

**Tarek Ben Hamdouch**

**AUTOMATIZACIÓN DEL DESPLIEGUE DE MODELOS DE INTELIGENCIA  
ARTIFICIAL HECHOS CON MONAI A AMAZON WEB SERVICES (AWS)**

**TRABAJO FIN DE GRADO**

**Dirigido per Jordi Massaguer Pla**

**Doble grado en Ingeniería Informática y en Biotecnología**



**UNIVERSITAT ROVIRA I VIRGILI**

**Tarragona**

**2024**



**Resumen.**

Este proyecto surgió ante la necesidad de facilitar el despliegue automatizado de modelos de inteligencia artificial para la inferencia en imágenes médicas utilizando servicios en la nube de Amazon Web Services (AWS). La inteligencia artificial está transformando el campo de la medicina, mejorando la eficiencia y precisión en el diagnóstico y tratamiento de enfermedades. En este contexto, MONAI (Medical Open Network for AI) proporciona una plataforma de código abierto para el desarrollo de modelos de IA en imágenes médicas, pero llevar estos modelos a producción sigue siendo un reto significativo.

Para realizar el proyecto se han utilizado GitHub Actions para automatizar y gestionar los servicios de AWS, permitiendo empaquetar los modelos y ejecutarlos sin necesidad de intervención humana. Además, se ha configurado el sistema para que realice la inferencia de manera autónoma cada vez que se sube una imagen a Amazon S3.

Esta automatización reduce el tiempo necesario para llevar un modelo desde la fase de desarrollo hasta la producción, garantizando consistencia y escalabilidad. El proyecto demuestra la eficacia y la viabilidad de esta prueba de concepto, permitiendo a los investigadores centrarse en la optimización de los modelos en lugar de en las tareas de despliegue, mejorando así la eficiencia y reduciendo los costos operativos.

**Resum.**

Aquest projecte va sorgir davant la necessitat de facilitar el desplegament automatitzat de models d'intel·ligència artificial per a la inferència en imatges mèdiques utilitzant serveis al núvol d'Amazon Web Services (AWS). La intel·ligència artificial està transformant el camp de la medicina, millorant l'eficiència i precisió en el diagnòstic i tractament de malalties. En aquest context, MONAI (Medical Open Network for AI) proporciona una plataforma de codi obert per al desenvolupament de models d'IA en imatges mèdiques, però portar aquests models a producció continua sent un repte significatiu.

Per realitzar el projecte s'han utilitzat GitHub Actions per automatitzar i gestionar els serveis d'AWS, permetent empaquetar els models i executar-los sense necessitat d'intervenció humana. A més a més, s'ha configurat el sistema perquè faci la inferència de manera autònoma cada vegada que es puja una imatge a Amazon S3.

Aquesta automatització redueix el temps necessari per portar un model des de la fase de desenvolupament fins a la producció, garantint consistència i escalabilitat. El projecte demostra l'eficàcia i la viabilitat d'aquesta prova de concepte, permetent als investigadors centrar-se en l'optimització dels models en lloc de les tasques de desplegament, millorant així l'eficiència i reduint els costos operatius.

**Abstract.**

This project came about in response to the need to facilitate the automated deployment of artificial intelligence models for medical image inference using Amazon Web Services (AWS). Artificial intelligence is transforming the field of medicine, improving its efficiency and accuracy in the diagnosis and treatment of diseases. In this context, MONAI (Medical Open Network for AI) provides an open source platform for the development of AI models in medical imaging, but bringing these models into production remains a significant challenge.

To implement the project, GitHub Actions has been used to automate and manage the AWS services, allowing the models to be packaged and executed without human intervention. In addition, the system has been configured to perform the inference autonomously each time an image is uploaded to Amazon S3.

This automation reduces the time required to take a model from development to production, ensuring consistency and scalability. The project demonstrates the effectiveness and viability of this proof of concept, allowing researchers to focus on model optimisation rather than deployment tasks, thus improving efficiency and reducing operational costs.

# Índice

<b>1</b>	<b>INTRODUCCIÓN .....</b>	<b>5</b>
1.1	OBJETIVOS.....	8
<b>2</b>	<b>PLANIFICACIÓN.....</b>	<b>9</b>
<b>3</b>	<b>DESARROLLO DEL PROYECTO.....</b>	<b>11</b>
3.1	REQUISITOS .....	11
3.2	DISEÑO .....	13
3.2.1	<i>Creación del MAP .....</i>	<i>14</i>
3.2.2	<i>Configuración de la inferencia.....</i>	<i>16</i>
3.2.3	<i>Ejecución de la inferencia.....</i>	<i>18</i>
3.3	IMPLEMENTACIÓN .....	19
3.3.1	<i>Creación del MAP .....</i>	<i>19</i>
3.3.2	<i>Configuración de la inferencia.....</i>	<i>22</i>
<b>4</b>	<b>RESULTADOS .....</b>	<b>25</b>
<b>5</b>	<b>EVALUACIÓN DE COSTOS .....</b>	<b>29</b>
<b>6</b>	<b>LEGISLACIÓN Y PROTECCIÓN DE DATOS .....</b>	<b>32</b>
<b>7</b>	<b>CONCLUSIONES .....</b>	<b>35</b>
<b>8</b>	<b>REFERENCIAS.....</b>	<b>36</b>
<b>APÉNDICE.....</b>		<b>37</b>
A	CONFIGURACIÓN AWS.....	37
	<i>Rol EC2 .....</i>	<i>37</i>
	<i>Usuario IAM.....</i>	<i>38</i>
	<i>HealthImaging.....</i>	<i>39</i>
	<i>Bucket S3.....</i>	<i>40</i>
	<i>Rol HealthImaging .....</i>	<i>40</i>
	<i>Rol lambda .....</i>	<i>41</i>
B	CONFIGURACIÓN GITHUB .....	42

## Índice de figuras

FIGURA 1 ESQUEMA DEL CICLO DEL DESARROLLO DE MODELOS CON MONAI.....	7
FIGURA 2 DIAGRAMA DE GANTT.....	9
FIGURA 3 DIAGRAMA CASOS DE USO .....	11
FIGURA 4 SERVICIOS DE AWS DONDE SE PUEDE EJECUTAR MONAI [2].....	13
FIGURA 5 DIAGRAMA CREACIÓN DEL MAP.....	15
FIGURA 6 ESQUEMA NOTIFICACIONES DE EVENTOS DE AMAZON S3 .....	16
FIGURA 7 DIAGRAMA CONFIGURACIÓN DE LA INFERENCIA .....	17
FIGURA 8 DIAGRAMA EJECUCIÓN DE LA INFERENCIA .....	18
FIGURA 9 INTERFAZ DE SELECCIÓN DE LES AMI EN LA CONSOLA DE AWS .....	19
FIGURA 10 AMIS QUE VIENEN CONFIGURADAS CON LOS DIVERS DE NVIDIA.....	20
FIGURA 11 RESULTADOS DE EJECUTA CREATE_MAP.YML .....	25
FIGURA 12 IMAGEN SUBIDA A AMAZON ECR.....	25
FIGURA 13 RESULTADOS DE EJECUTAR CONFIGURE_INFERENCE.YML .....	26
FIGURA 14 SUBIR FICHEROS DE PRUEBA.....	26
FIGURA 15 RESULTADOS GUARDADOS EN S3.....	27
FIGURA 16 RESULTADOS EN HEALTHIMAGING.....	27
FIGURA 17 CONFIGURACIÓN FLEXVIEW .....	28
FIGURA 18 VISUALIZACIÓN DE UN DICOM EN FLEXVIEW.....	28
FIGURA 19 FACTURA AWS FEBRERO.....	29
FIGURA 20 EJEMPLO DE NOTIFICACIÓN CUANDO SE ACTIVA LA ALARMA .....	29
FIGURA 21 FACTURA AWS ABRIL.....	30
FIGURA 22 FACTURA AWS MAYO.....	30

## Índice de códigos

CÓDIGO 1. SUSTITUCIÓN DE LAS VARIABLES Y ENVÍO DEL SCRIPT A LA INSTANCIA EC2 .....	22
CÓDIGO 2. CREACIÓN DE LA FUNCIÓN LAMBDA.....	23
CÓDIGO 3. CONFIGURACIÓN DE LA NOTIFICACIÓN .....	24

## **Abreviaciones**

AMI - Amazon Machine Image  
AWS - Amazon Web Services  
CLI - Command Line Interface  
CI/CD - Continuous Integration/Continuous Deployment  
DICOM - Digital Imaging and Communication In Medicine  
EBS - Elastic Block Store  
EC2 - Elastic Compute Cloud  
ECR - Elastic Container Registry  
IAM - Identity and Access Management  
IA - Intel·ligència Artificial  
MAP - MONAI Application Package  
MONAI - Medical Open Network for AI  
SSM - Simple Systems Manager  
S3 - Simple Storage Service  
SNS - Simple Notification Service  
SQS - Simple Queue Service



## 1 Introducción

La inteligencia artificial (IA) está transformando el campo de la medicina de modos que hace pocos años eran inimaginables. Su capacidad para procesar grandes volúmenes de datos con precisión y velocidad está permitiendo avances significativos en el diagnóstico y tratamiento de enfermedades. Las herramientas de IA, como Mia<sup>1</sup>, están demostrando ser especialmente valiosas en la detección precoz de cánceres y otras enfermedades, identificando signos que pueden pasar desapercibidos para los médicos. Esta tecnología no sólo aumenta la tasa de detección, sino también reduce el tiempo de espera para los resultados, mejorando así la calidad de la atención a los pacientes y aumentando sus posibilidades de supervivencia. Además, la IA puede ayudar a aliviar la carga de trabajo de los profesionales de la salud, permitiéndoles dedicar más tiempo a la interacción directa con los pacientes ya la toma de decisiones.

El creciente uso de la inteligencia artificial (IA) en el sector sanitario ha dado lugar a la creación de herramientas especializadas para el análisis de imágenes médicas. Una de las plataformas más innovadoras e influyentes en este ámbito es MONAI (Medical Open Network for AI). El proyecto, desarrollado inicialmente por NVIDIA juntamente con King's College de Londres, ofrece un marco de trabajo de código abierto basado en PyTorch, en el que los investigadores pueden acelerar la investigación y la colaboración clínica en imágenes médicas.

Lanzada por primera vez en 2019, MONAI ofrece una plataforma que es fácil de utilizar, proporciona resultados reproducibles y está optimizada para las necesidades específicas de los datos médicos, capaz de gestionar formatos únicos, resoluciones y metainformación especializada de imágenes médicas. Uno de sus objetivos clave es permitir la reproducibilidad de los experimentos, de modo que los investigadores puedan compartir y comparar resultados. Esto es especialmente importante para poder validar y perfeccionar los modelos.

MONAI se divide en tres componentes clave que ofrecen funcionalidades específicas para distintas necesidades de los investigadores y profesionales de la salud. Estas herramientas permiten cubrir todo el proceso para el desarrollo de modelos, desde la investigación hasta la producción clínica.

1. **MONAI Label** es una herramienta inteligente de etiquetado de imágenes que utiliza asistencia de IA para reducir el tiempo y el esfuerzo de anotar nuevos conjuntos de datos.
2. **MONAI Core** es la biblioteca principal del Proyecto MONAI y proporciona herramientas y funcionalidades específicas para procesar los datos y entrenar modelos de IA para imágenes médicas.

---

<sup>1</sup> Mammography Intelligent Assessment es una plataforma de IA para la detección de cáncer de mama.

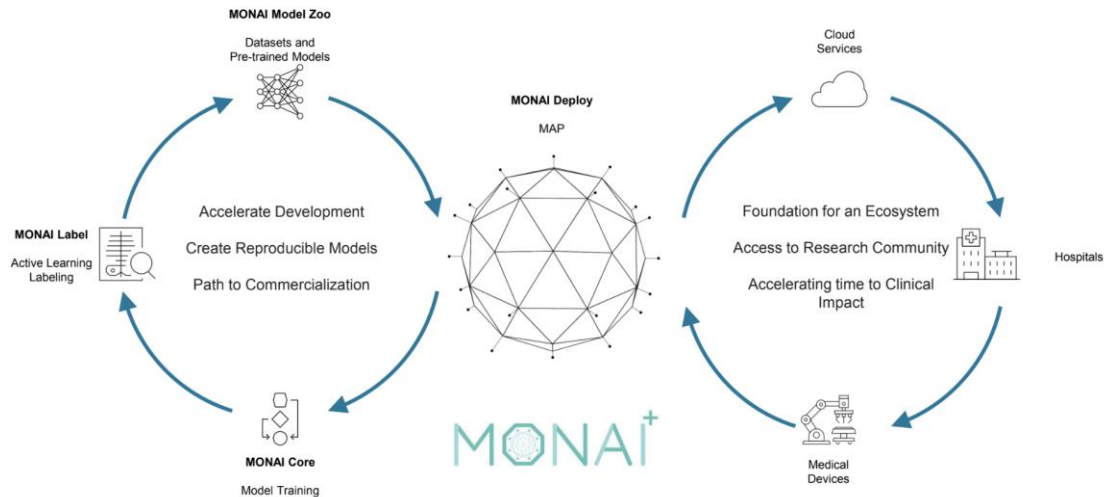
3. **MONAI Deploy** tiene como objetivo empaquetar, probar, desplegar y ejecutar aplicaciones de IA en producción clínica.

En los últimos años, la inteligencia artificial ha demostrado tener un potencial transformador en distintos sectores. Sin embargo, grande parte de los proyectos de IA no logran llegar a la producción, quedándose en fases iniciales como pruebas piloto. Según la última encuesta realizada por KDnuggets [1], un 80% de los modelos de inteligencia artificial nunca llegan a la producción. Automatizar este proceso no sólo mejora la eficiencia, sino que también garantiza consistencia, escalabilidad y seguridad, permitiendo que más modelos pasen del laboratorio a la práctica real y aportan un valor tangible a las empresas e instituciones.

Automatizar el despliegue de las IA permite reducir el tiempo necesario para quitar un modelo desde la fase de desarrollo hasta la producción. Un despliegue manual puede ser lento y sujeto a errores humanos, mientras que la automatización asegura una transición más rápida y eficiente. Además, el despliegue en la nube es especialmente importante porque permite el acceso a varios servicios y recursos sin necesidad de mantener hardware físico. Esto facilita la escalabilidad, flexibilidad y capacidad de ejecutar modelos de IA en entornos distribuidos, mejorando así la eficiencia y reduciendo los costes operativos.

La nube ofrece una infraestructura robusta y adaptable que permite gestionar grandes volúmenes de datos y procesarlos con alta velocidad. Esto es crucial para los proyectos de IA, puesto que a menudo requieren recursos computacionales intensivos que no son fácilmente accesibles a través de infraestructuras locales. Utilizando servicios en la nube, las organizaciones pueden acceder a recursos bajo demanda, pagando sólo lo que utilizan, lo que optimiza los gastos y evita inversiones elevadas en hardware y mantenimiento.

Amazon Web Services (AWS) es la empresa líder en servicios de nube, ofreciendo una amplia gama de servicios que cubren desde el almacenamiento de datos hasta el procesamiento y el análisis avanzado. Su infraestructura global y sus soluciones de seguridad robustas hacen que AWS sea una opción preferida para empresas que buscan desplegar y gestionar modelos de IA de forma eficiente y segura. AWS proporciona herramientas específicas como Amazon SageMaker para entrenar modelos de IA, permitiendo a los desarrolladores crear, entrenar y desplegar modelos. Además, AWS dispone de servicios como Amazon HealthLake y Amazon HealthImaging, diseñados para gestionar y analizar datos médicos, ofreciendo soluciones avanzadas para la salud digital y mejorando la capacidad de las instituciones sanitarias para gestionar grandes volúmenes de información de forma eficiente y segura.



**Figura 1** Esquema del ciclo del desarrollo de modelos con MONAI

El esquema ilustra el ciclo completo del desarrollo de modelos de IA médica con MONAI, desde la fase inicial de desarrollo del modelo hasta el despliegue clínico. En el centro del esquema se encuentra el MONAI Application Package (MAP), que sirve como punto de conexión clave entre estas dos fases. El que queremos hacer es que una vez tenemos el modelo ya está entrenado, empaquetarlo en un MAP para facilitar su despliegue en un entorno clínico. El MAP consiste en una imagen de un contenedor docker con el modelo entrenado que contiene una aplicación ejecutable que dada una entrada, hace la inferencia con el modelo para generar las salidas.

Para conseguir automatizar el proceso usaremos las GitHub Actions. GitHub Actions es una plataforma de integración y despliegue continuos (CI/CD) que permite automatizar los procesos de construcción, prueba y despliegue de proyectos. Con GitHub Actions, se pueden crear flujos de trabajo que se desencadenan por acontecimientos específicos, como por ejemplo cuando alguien sube código al repositorio (push) o cuando crea una solicitud de incorporación de cambios (pull request).

He elegido GitHub Actions porque GitHub es la plataforma de control de versiones más popular y ampliamente utilizada por desarrolladores de todo el mundo, facilitando la gestión del código fuente de manera colaborativa y segura. Esta popularidad la hace una opción ideal para integrar automatización puesto que, al tener una base de usuarios muy alta, existe una gran cantidad de recursos, documentación y comunidades de apoyo. Además, permite ejecutar los flujos de trabajo (workflows) de manera gratuita para los repositorios públicos, lo cual resulta especialmente beneficiosa para proyectos de código abierto.

## 1.1 Objetivos

El objetivo principal de este proyecto es hacer una prueba de concepto para automatizar el proceso de despliegue de modelos desarrollados con MONAI a la nube, facilitando que los investigadores puedan centrarse en la optimización de los modelos en lugar de las tareas de despliegue. Esta automatización tiene que mejorar la eficiencia del proceso, reduciendo el tiempo y los costes asociados al despliegue manual.

Este proyecto está dirigido principalmente a ingenieros de aprendizaje automático (machine learning), así como a los ingenieros de plataformas que se dedican a crear infraestructuras a la nube. La automatización del despliegue a partir del código fuente incluye la creación de la estructura necesaria a la nube, haciendo el proceso más corto y simplificado, y abaratando los costes asociados. Esto permitirá que aquellos profesionales con gran conocimiento en aprendizaje automático, pero sin experiencia en tecnologías de nube, puedan aprovechar las herramientas y servicios disponibles para desplegar modelos automáticamente, maximizando así su productividad.

En el ámbito académico, el objetivo es expandir el conocimiento y adquirir habilidades prácticas que permitan afrontar los retos tecnológicos actuales. En la actualidad, muchas empresas e instituciones están migrando a soluciones basadas en la nube para aprovechar su flexibilidad, escalabilidad y eficiencia en costes.

Durante mi carrera, no he tenido la oportunidad de trabajar con tecnologías de nube, y por tanto he querido aprovechar esta ocasión para aprender más sobre cómo funcionan. Esto incluye entender qué servicios ofrecen las principales plataformas de nube, como por ejemplo AWS, el coste asociado en la contratación de estos servicios, y como pueden ser utilizados para desplegar modelos de inteligencia artificial desarrollados con MONAI.

El conocimiento adquirido en este proceso no solo me permitirá comprender mejor las ventajas y desafíos de trabajar con la nube, sino que también me dotará de las habilidades necesarias para aplicar estas tecnologías en proyectos futuros. Así, estaré mejor preparado para afrontar los retos tecnológicos de mi carrera profesional.

## 2 Planificació

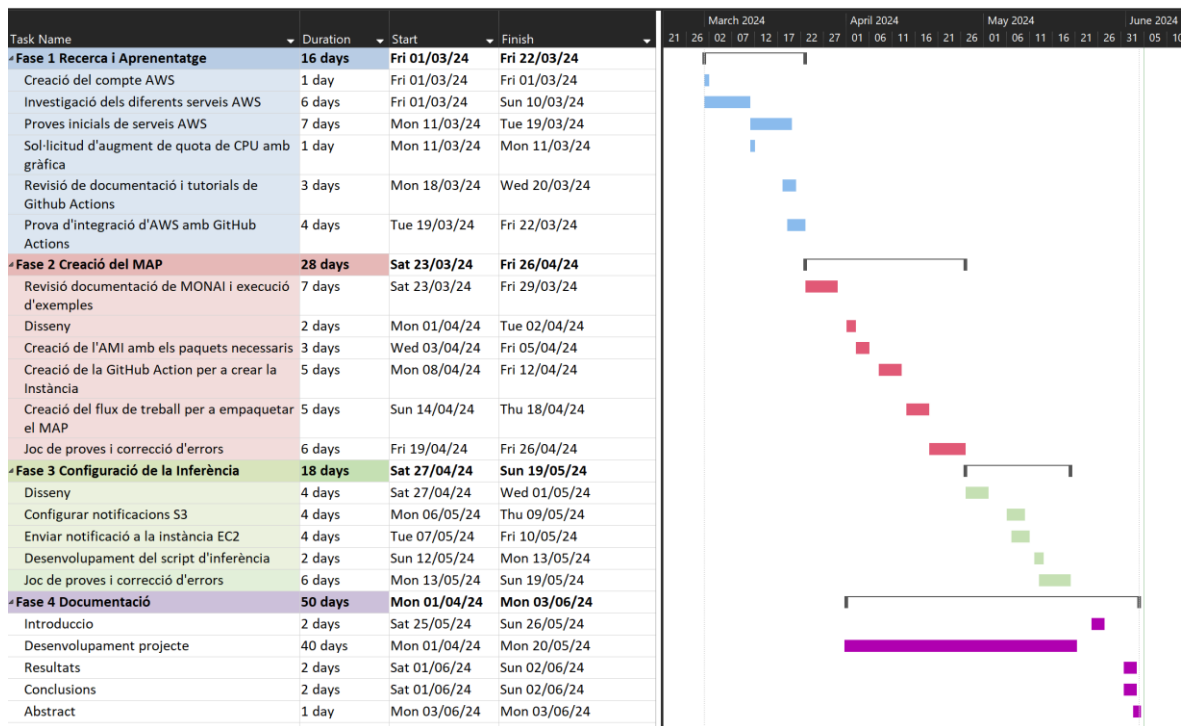


Figura 2 Diagrama de Gantt

### Fase 1 Recerca i Aprendizaje

En esta primera fase, se empezó con la creación de la cuenta de AWS, esencial para acceder a los varios servicios a la nube que ofrece la plataforma. Se procedió a investigar y explorar los diferentes servicios que proporciona AWS, con el objetivo de entender como podrían ser utilizados para el proyecto. Esto incluyó la realización de pruebas iniciales con estos servicios para familiarizarse con su funcionamiento y sus capacidades. Además, se solicitó un aumento de cuota de CPU con gráfica, necesario para trabajar con instancias que incluyan GPU. Paralelamente, se revisó la documentación y los tutoriales de GitHub Actions para comprender como podrían ser integrados con AWS para automatizar los procesos. Finalmente, se realizaron pruebas de integración entre AWS y GitHub Actions para asegurarse que las herramientas funcionaban correctamente, utilizando un usuario IAM y configurando la AWS CLI.

### Fase 2 Creación del MAP

En esta etapa, se empezó con una revisión de la documentación de MONAI y la ejecución de ejemplos para comprender como funcionaba. A continuación, se procedió al diseño, identificando los paquetes y configuraciones necesarias para crear una imagen de máquina (AMI) con todos los componentes requeridos. Una vez identificados los requisitos, se creó el AMI con los paquetes necesarios, incluyendo los controladores de NVIDIA, Docker y otras herramientas esenciales. Después, se desarrolló una GitHub Action para automatizar la creación de la instancia EC2, seguido de la creación de un flujo de trabajo que permitiera empaquetar el MAP de manera eficiente. Finalmente, se llevó a cabo un juego de pruebas exhaustivo para asegurarse que todo funcionaba correctamente, corrigiendo cualquier error que se presentara durante el proceso.

### **Fase 3 Configuración de la Inferencia**

En esta fase, se empezó con el diseño de la solución, incluyendo la configuración de notificaciones de S3 para asegurar que cada vez que se sube una imagen, se genere una alerta automática. Se configuró el envío de estas notificaciones a la instancia EC2 para que esta pueda procesarlas. A continuación, se desarrolló el script de inferencia que descarga los datos, ejecuta el modelo y sube los resultados. Para acabar, se llevó a cabo un juego de pruebas exhaustivo para verificar que todo funcionaba correctamente y se corrigieron los errores identificados para garantizar una operación fluida y precisa.

### **Fase 4 Documentación**

Durante esta fase final, se redactó toda la documentación necesaria para el proyecto. Esto incluye una introducción detallada, el desarrollo del proyecto, los resultados obtenidos y las conclusiones. La documentación asegura que cualquier persona que quiera replicar o entender el proyecto tenga toda la información necesaria de manera clara y organizada.

### 3 Desarrollo del proyecto

#### 3.1 Requisitos

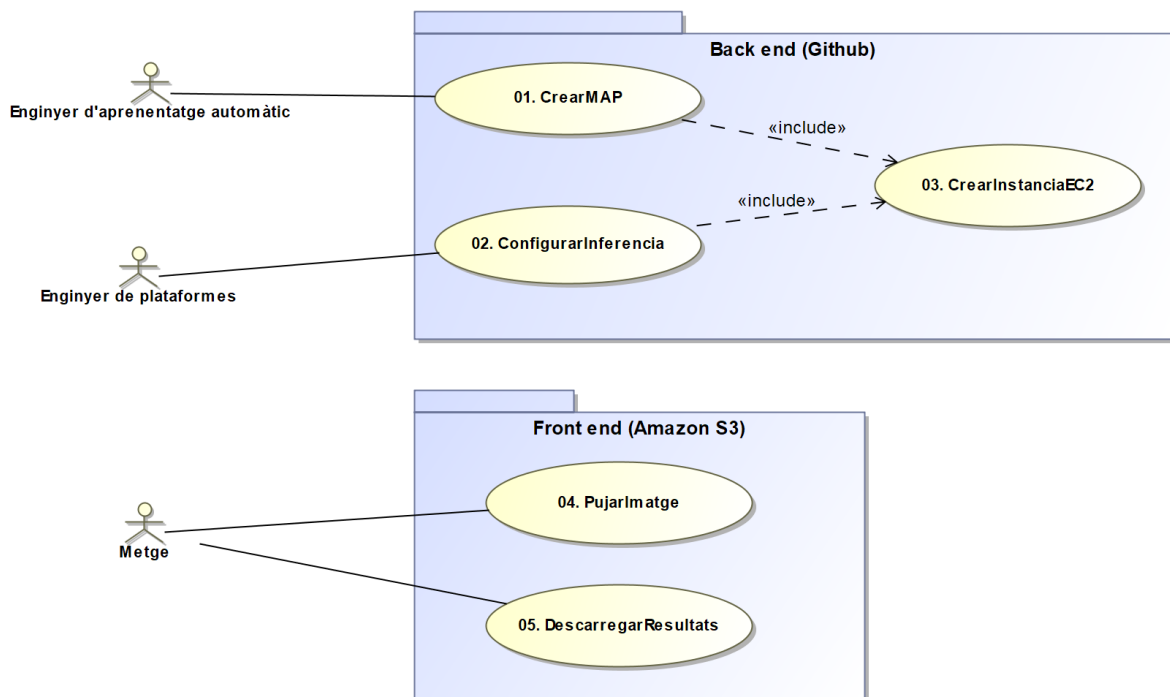
El sistema tiene que permitir el despliegue automático del modelo de aprendizaje automático y la configuración del proceso de inferencia. Esto implica la instalación y puesta en marcha del modelo en uno en torno a producción, así como el establecimiento de las configuraciones necesarias para su operación continuada.

El sistema tiene que utilizar los servicios de Amazon Web Services (AWS) para el despliegue y ejecución del modelo de machine learning. Esto incluye, pero no se limita a, servicios como Amazon S3 para el almacenamiento de datos, AWS Lambda para la computación sin servidores, Amazon SageMaker para la gestión del machine learning, y otros servicios relevantes de AWS.

El sistema tiene que utilizar GitHub como plataforma para el control de versiones y la colaboración en el código. Esto permitirá la gestión centralizada del código fuente, la colaboración entre los desarrolladores, y la integración continua y despliegue continuo (CI/CD).

Restricciones:

- Solo se desarrollará el backend, sin incluir componentes de frontend.
- El sistema tiene que ser fácil de configurar, pensando en el uso por parte de un ingeniero con conocimientos de machine learning pero sin experiencia previa en AWS.



**Figura 3** Diagrama casos de uso

El proyecto se estructura en dos componentes principales: el backend y lo frontend. El backend, gestionado principalmente con GitHub Actions, está centrado en la creación y configuración del MAP, así como en la instancia EC2 que ejecuta la inferencia. Esta parte del proyecto ha sido nuestro foco principal, garantizando que los procesos de automatización y despliegue funcionen de manera eficiente y sin necesidad de intervención humana.

Para hacer esta prueba de concepto, se utilizará directamente Amazon S3. Sin embargo, hay que remarcar que, en una versión definitiva del proyecto, sería necesario implementar una interfaz web que permitiera a los usuarios finales, como los médicos, interactuar con el sistema de manera más intuitiva y amigable. Esta interfaz web tendría que permitir la subida de imágenes y la visualización o descarga de los resultados sin tener que acceder directamente a S3, mejorando así la experiencia de usuario y asegurando una mayor seguridad y control sobre los datos.



### 3.2 Diseño



**Figura 4** Servicios de AWS donde se puede ejecutar MONAI [2]

En la parte del despliegue, vemos que tenemos dos alternativas, se puede ejecutar tanto en Elastic Computo Cloud (EC2) como en Amazon Sagemaker.

Amazon EC2 es un servicio que proporciona capacidad de computación escalable a la nube. Está diseñado para facilitar a los desarrolladores una infraestructura a la nube segura y escalable que reduce los costes de tener maquinaria física. Amazon EC2 permite a los usuarios alquilar máquinas virtuales donde pueden ejecutar sus aplicaciones, ofreciendo la flexibilidad de configurarlas según las necesidades específicas de cada proyecto, puesto que en la hora de crear una instancia, se puede configurar porque la máquina virtual se ejecute en varios entornos o con características específicas.

Amazon SageMaker es un servicio que reúne varias herramientas que permiten a los desarrolladores y científicos de datos crear, entrenar y desplegar modelos de aprendizaje automático (machine learning). SageMaker simplifica cada paso del proceso, desde la preparación de datos hasta el entrenamiento y la inferencia, proporcionando herramientas integradas que facilitan el desarrollo y despliegue de modelos de IA. Además, facilita el despliegue de modelos entrenados en entornos de producción, ofreciendo inferencia en tiempo real y permitiendo a los usuarios escalar sus servicios de IA según las necesidades.

En un principio nos planteamos hacer el proyecto en sagemaker, pero nos dimos cuenta que nos iría más bien hacerlo en EC2. Hay muchos ejemplos de como desplegar los modelos en Sagemaker, y todos usan jupyter notebooks. Los Jupyter Notebooks son una herramienta de código abierto que permite a los usuarios crear y compartir documentos que contienen código ejecutable, visualizaciones y texto explicativo todo en un mismo fichero.

Tal como hemos comentado en la introducción, el que queremos hacer es automatizar el despliegue usando como intermediario un MAP (veáis figura 1) y, por lo tanto, no nos sirve el sagemaker porque en vez de usar el MAP, usa los modelos directamente para hacer la inferencia.

### 3.2.1 Creación del MAP

El primer paso para pasar los modelos a producción es empaquetar la aplicación junto con el modelo en un MAP, esta imagen de Docker cumple con una serie de especificaciones que aseguran que la aplicación contenida sea fácilmente portable asegura que se puede ejecutar en diferentes entornos sin necesidad de modificaciones.

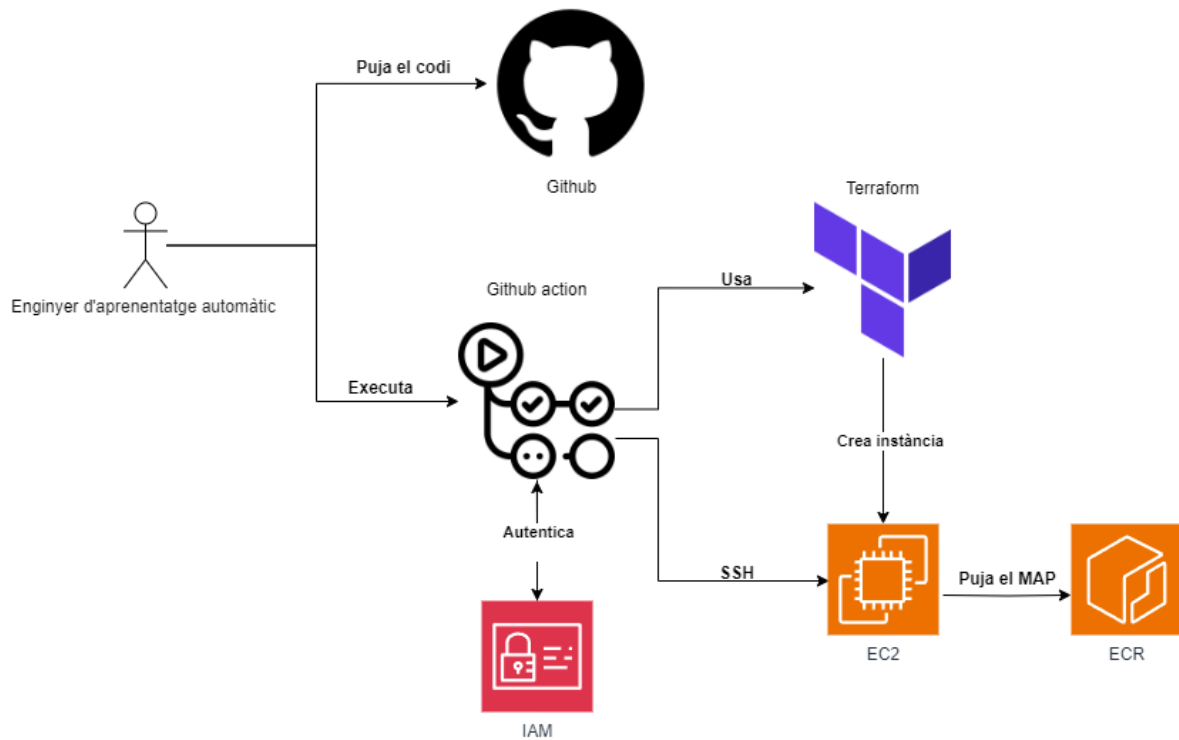
Para crear el MAP es necesario disponer de una tarjeta gráfica compatible con NVIDIA CUDA. Dado que el portátil desde donde trabajo no es compatible, y otros muchos programadores probablemente tampoco disponen de este recurso, he decidido automatizar la creación del MAP mediante la reserva de una instancia con GPU a EC2. Esta solución permitirá a los desarrolladores crear MAPs de manera eficiente y sin las limitaciones de maquinaria.

Para reservar una instancia a EC2 se puede hacer mediante pedidos de la línea de mandos (CLI) de AWS. Sin embargo, he decidido utilizar Terraform. El uso de Terraform permite hacer configuraciones de manera más sencilla y reutilizable, puesto que solo hay que modificar el fichero main.tf para ajustar las configuraciones necesarias. El más importante de este fichero de configuración es el tipo de instancia y la Imagen de Máquina de Amazon (AMI).

Como que necesitamos una instancia con gráfica tenemos que elegir entre instancias:

- P3 tienen hasta 8 GPU NVIDIA Tesla V100.
- P4 tienen hasta 8 GPU NVIDIA Tesla A100.
- G3 tienen hasta 4 GPU NVIDIA Tesla M60.
- G4 tienen hasta 4 GPU NVIDIA T4.
- G5 tienen hasta 8 GPU NVIDIA A10G.

Las AMI son imágenes preconfiguradas que proporcionan la información necesaria para lanzar una instancia EC2. Una AMI incluye el sistema operativo, el servidor de aplicaciones y las aplicaciones necesarias para el funcionamiento de la instancia. Las AMI permiten a los usuarios crear copias exactas de sus configuraciones y desplegar múltiples instancias con la misma configuración en cuestión de minutos. Esto facilita la gestión y la escalabilidad de los servicios a la nube, puesto que los usuarios pueden seleccionar una AMI específica para adaptarse a sus necesidades de cálculo y software. Además, las AMI pueden ser personalizadas para incluir aplicaciones específicas, controladores, configuraciones de seguridad.



**Figura 5** Diagrama Creación del MAP

El diagrama ilustra el proceso de creación del MAP. El proceso empieza cuando el ingeniero de aprendizaje automático sube el código al repositorio de GitHub. A continuación, se ejecuta el flujo de trabajo que primero se autentica a AWS mediante credenciales IAM para tener acceso a los servicios.

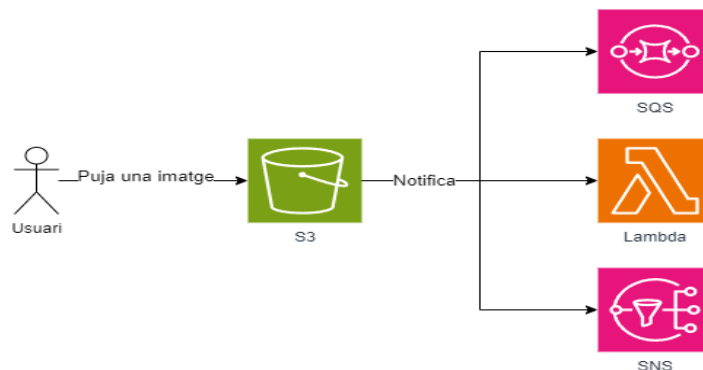
Una vez autenticada, el flujo de trabajo utiliza Terraform para gestionar y crear una instancia EC2. Esta máquina virtual está configurada con los paquetes y herramientas requeridos para empaquetar el modelo.

Después, la GitHub Action se conecta a la instancia EC2 vía SSH para crear y subir el MAP a un registro de contenedores ECR (Elastic Container Registry).

### 3.2.2 Configuración de la inferencia

En esta parte, el que queremos hacer es crear una nueva instancia que haga automáticamente la inferencia cada vez que alguien cuelgue una nueva imagen o archivo ZIP en los contenedores de S3. Esto implica configurar un sistema donde la instancia EC2 procesa los datos y genera resultados de manera autónoma. Los resultados obtenidos de la inferencia se subirán al mismo contenedor de S3 para ser descargados por los usuarios, garantizando un acceso rápido y fácil a los resultados. Además, se importarán a HealthImaging, donde se podrán analizar los metadatos de las imágenes Digital Imaging and Communication In Medicine (DICOM) y visualizarlas posteriormente. Este enfoque permite que la inferencia se realice de manera continua y eficiente, asegurando que los nuevos datos se procesan inmediatamente después de ser colgadas, proporcionando respuestas rápidas a los usuarios.

En la primera parte, creo una instancia para hacer el MAP y ahora, creo otra instancia para ejecutar el MAP y hacer la inferencia. La justificación de hacerlo en dos máquinas diferentes es la siguiente. Una vez configurada la segunda máquina, esta simplemente ejecuta la última imagen subida al registro ECR. Esto significa que podemos actualizar el modelo y subirlo al ECR tantas veces como sea necesario sin tener que reconfigurar la máquina de inferencia. Esta separación de tareas hace que el proceso sea más modular y mantiene la máquina de inferencia transparente respecto a las actualizaciones del modelo. Así, cualquier cambio o mejora en el modelo se puede desplegar rápidamente y de manera eficiente, sin interrumpir el flujo de trabajo de la inferencia.



**Figura 6** Esquema notificaciones de eventos de Amazon S3

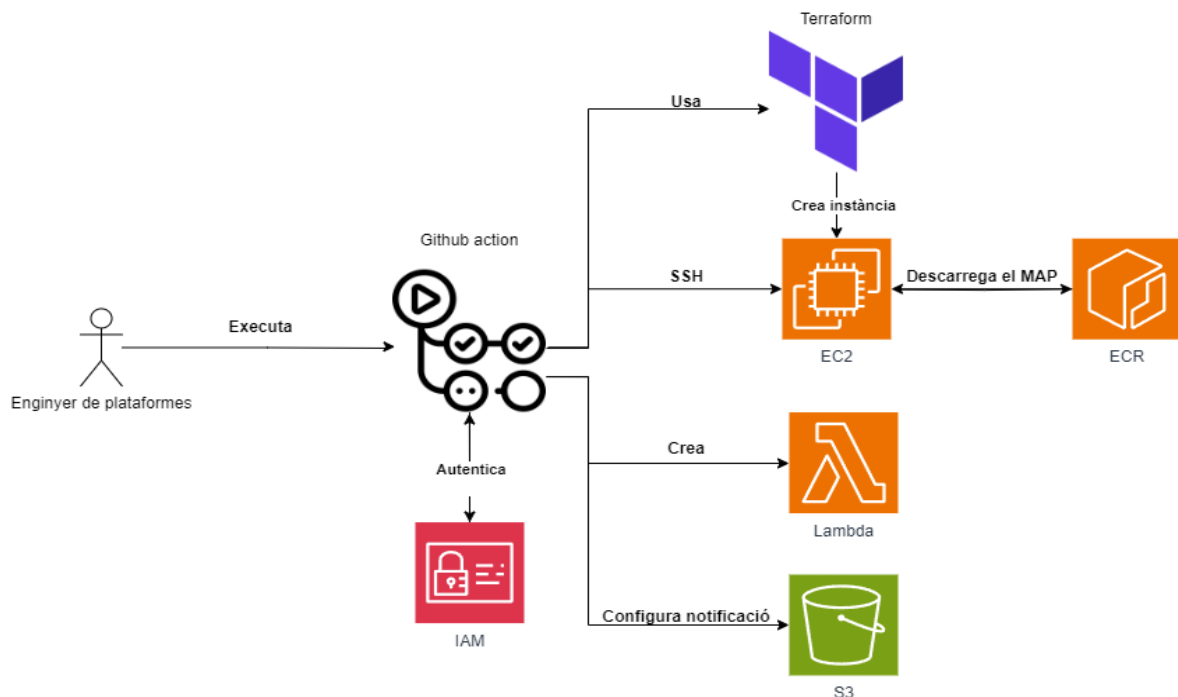
Con la máquina ya creada, tenemos que hacer que de alguna manera cada vez que se suba una imagen al contenedor de S3, se descargue el fichero y ejecute el modelo. En S3, se puede configurar para que envíe notificaciones cuando alguien crea un objeto de tres formas diferentes:

- **Función Lambda:** Las funciones Lambda son fragmentos de código que se ejecutan en respuesta a acontecimientos y permiten la ejecución automática de tareas sin necesidad de gestionar servidores. Esto significa que, cada vez que se sube una imagen a un contenedor S3, se puede activar automáticamente una función Lambda que se encargue de descargar el fichero subido y de ejecutar el modelo. Utilizar Lambda por este propósito proporciona una solución escalable y eficiente, puesto que las funciones Lambda solo se ejecutan cuando se producen los acontecimientos especificados, reduciendo la necesidad de recursos permanentes y permitiendo una respuesta rápida y automatizada en las subidas de ficheros.

- **Servicio Simple de Notificaciones (SNS):** Es un servicio de mensajería basado en el modelo productor/subscriptor, donde los mensajes son publicados en un tema (topic) y todos los subscriptores de este reciben las notificaciones. Los subscriptores pueden ser otros servicios, aplicaciones o personas que necesitan recibir el aviso. Este enfoque facilita el envío de mensajes a múltiples destinatarios simultáneamente, asegurando que todos los servicios necesarios reciban la notificación permiten una comunicación asíncrona y escalable.

• **Servicio de Cola Simple (SQS):** Las notificaciones se pueden enviar a una cola SQS, que serán leídas por un servidor. Amazon SQS es un servicio de mensajería completamente gestionado que permite desconectar y escalar componentes distribuidos, sistemas y aplicaciones. Al enviar las notificaciones a una cola SQS, se asegura que cada mensaje (como por ejemplo la subida de una nueva imagen) se procese de manera secuencial. Un servidor puede leer los mensajes de la cola, descargar las imágenes subidas y ejecutar el modelo correspondiente. Este enfoque permite gestionar el procesamiento de las imágenes de manera ordenada y fiable, asegurando que no se pierdan mensajes y que se procesen en la orden en que han llegado.

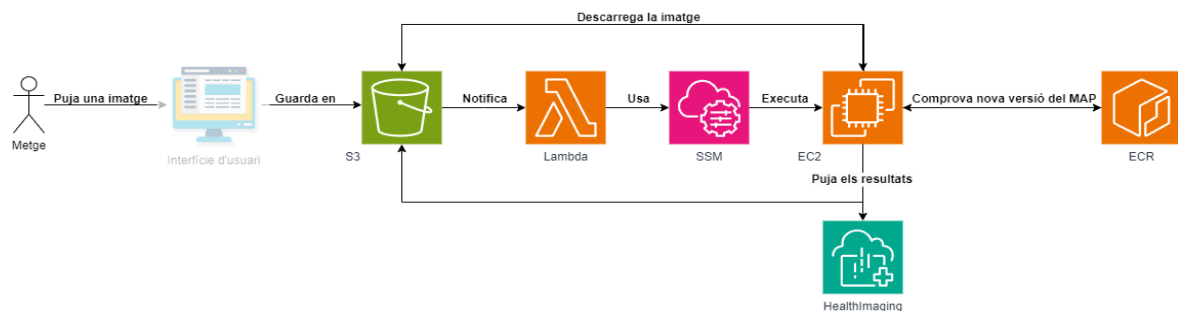
Al final me he decidido por la primera opción. Para recibir los mensajes SNS, tienes que dejar la máquina continuamente escuchando el puerto, cosa que puede resultar costoso e ineficiente. En las colas, los mensajes se almacenan hasta que un servidor los lee, cosa que puede requerir mantener el servidor activo todo el tiempo o manualmente ir parando y poniendo en marcha la máquina. En cambio, usando una función Lambda podemos poner en marcha y parar la instancia solo cuando sea necesario, reduciendo así los costes de tener la máquina siempre activa. Esta solución es más eficiente económicamente, puesto que las funciones Lambda solo se facturan por el tiempo real de ejecución y permiten una respuesta inmediata en las subidas de ficheros.



**Figura 7** Diagrama configuración de la inferencia

El diagrama muestra el proceso de configuración de la inferencia. Todo empieza cuando el ingeniero de plataformas ejecuta el flujo de trabajo. El primer paso de crear la instancia es muy parecido a la creación del MAP, pero en este caso, en vez de subir la imagen, la descargamos desde el ECR. A continuación, se crea una función Lambda que será la encargada de iniciar la inferencia. Finalmente, se configura S3 para que notifique a la función Lambda cada vez que se sube una nueva imagen. Esta configuración permite que la función Lambda active la instancia EC2 para descargar la imagen del S3 y ejecutar el proceso de inferencia automáticamente, asegurando una respuesta rápida en las subidas de ficheros.

### 3.2.3 Ejecución de la inferencia



**Figura 8** Diagrama ejecución de la inferencia

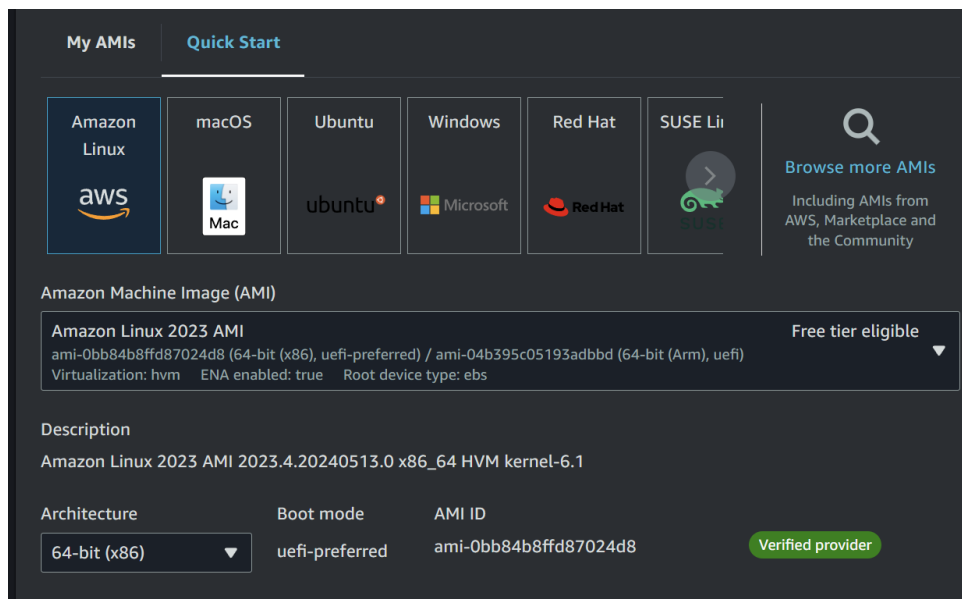
El diagrama muestra el proceso de ejecución de la inferencia. El proceso empieza cuando el médico sube una imagen. Idealmente, este paso se realizaría a través de una interfaz de usuario que facilitaría la interacción, pero para las pruebas, las imágenes se subirán directamente a S3. Una vez la imagen está guardada en S3, el servicio genera una notificación y pone en marcha una función Lambda.

La función Lambda utiliza el servicio AWS Systems Manager (SSM) para iniciar la instancia EC2 y ejecutar la script de inferencia. La instancia EC2 descarga la imagen desde el S3 y comprueba si hay una nueva versión del MAP a ECR. Una vez completada la inferencia, los resultados se suben a HealthImaging para su posterior análisis.

### 3.3 Implementación

#### 3.3.1 Creación del MAP

Tal como recomiendan desde MONAI, he utilizado la instancia g4dn.xlarge, que dispone de una tarjeta gráfica NVIDIA T4 de 16 GB junto con 4 vCPU y 16 GB de RAM, y que tiene un coste de 0,526 dólares por hora. Esta configuración proporciona la potencia de cálculo suficiente para poder empaquetar y ejecutar los modelos. Es muy importante tener en cuenta que, si es la primera vez que utilizas máquinas con GPU en tu cuenta de AWS, tienes que solicitar un aumento de la cuota para obtener permiso para ejecutarlas, puesto que por defecto están restringidas.



**Figura 9** Interfaz de selección de les AMI en la consola de AWS

Inicialmente, creé una instancia con la imagen de Ubuntu y configuré todos los paquetes necesarios para generar los MAP. La imagen de Ubuntu por defecto es una imagen muy ligera que solo contiene los componentes básicos para funcionar. Para preparar el entorno, tuve que seguir varios pasos detallados:

- **Instalación de los controladores NVIDIA:** Como que la instancia necesita apoyo para GPU, instalé los controladores NVIDIA para asegurarme que la tarjeta gráfica fuera reconocida y utilizada correctamente por el sistema.
- **Instalación de Docker:** Docker es esencial para la creación y ejecución de contenedores.
- **Instalación de la AWS CLI:** La herramienta de línea de mandos de AWS es necesaria para gestionar los servicios de AWS desde la instancia.
- **Instalación de Anaconda:** Para crear uno en torno a desarrollo Python virtual.

- Configuración del entorno virtual de Python: Con Anaconda instalado, creé un entorno virtual dedicado. Dentro de este entorno, instalé las librerías requeridas, como por ejemplo MONAI, PyTorch y otras dependencias necesarias para empaquetar y ejecutar los modelos.

Todos estos pasos los incluí en el script `ami.sh` para automatizar el proceso de configuración de la instancia. Para evitar tener que reinstalar y configurar todo cada vez que se crea una nueva instancia, cosa que supondría una gran pérdida de tiempo, creé una AMI a partir de una instancia ya configurada. EC2 permite crear AMI a partir de instancias existentes, capturando así todo el entorno de configuración e instalaciones necesarias.

A pesar de que el uso de una AMI personalizada es una solución eficiente para ahorrar tiempo y asegurar la consistencia en la configuración de las instancias, presenta algunos problemas, entre los cuales destaca la disponibilidad regional. Las AMI son un recurso regional, lo cual significa que solo están disponibles en la región desde donde se crearon y compartir. Para hacer que una AMI esté disponible en una región diferente, hay que copiar la AMI en la región deseada y, posteriormente, compartirla. Este proceso implica costes por almacenamiento y transferencia de datos.

Además, otro de los problemas significativos es el mantenimiento y la actualización de las AMI. Mantener una AMI actualizada con las últimas revisiones de seguridad y versiones de paquetes puede ser una tarea desafiando. Para solucionar esto, se puede utilizar AWS Systems Manager Automation. Esta herramienta permite aplicar revisiones de forma automática a una AMI con las versiones más recientes de los paquetes especificados.



**Figura 10** AMIs que vienen configuradas con los drivers de NVIDIA

En el catálogo de AMI se pueden encontrar varias imágenes que ya vienen preparadas con los controladores de NVIDIA instalados junto con Docker. Como que son imágenes oficiales, Amazon se encarga de mantenerlas actualizadas y seguras con las actualizaciones más recientes. Sin embargo, una vez puesta en marcha esta imagen, todavía tenemos que acabar de instalar las librerías de Python necesarias para nuestro proyecto.



Una vez aclaradas las características de las máquinas virtuales que reservamos, tenemos que hablar del flujo de trabajo que seguimos para crear el MAP. Este es el proceso detallado:

1. El primer paso para crear el MAP es autenticarnos con las credenciales de AWS. Esto se puede hacer utilizando la AWS CLI, que ya está instalada por defecto en las máquinas virtuales de GitHub, o gritando a la acción `configure-aws-credentials`. En cualquier de los dos casos, no se tienen que escribir las credenciales directamente en el código, sino que se tienen que guardar en los secretos del repositorio. Los secretos son variables que permitan almacenar información confidencial y que se encripten antes de llegar a GitHub. Además, los ejecutores alojados por GitHub ejecutan el código dentro de máquinas virtuales efímeras y aisladas. Esto asegura que no hay manera de comprometer persistentemente este entorno ni acceder a más información otros usuarios, puesto que una vez finaliza el proceso, la máquina virtual se "limpia". Esta característica proporciona una capa adicional de seguridad, garantizando que no se pueden reutilizar ni comprometer los datos o claves temporales utilizadas durante la ejecución del código.
2. El segundo paso es generar una clave RSA temporal para poder conectarse a la instancia que crearemos intermediando SSH.
3. El tercer paso es ejecutar Terraform para crear la instancia, asegurándose de pasar la clave pública creada anteriormente.
4. Una vez creada la instancia, le asignamos un rol. Los roles en AWS son una manera segura de conceder permisos para acceder a otros servicios de AWS sin necesidad de utilizar credenciales de usuario. Los roles se definen a través del servicio AWS Identity and Access Management (IAM) y se asignan a las instancias para permitir que actúen con los permisos especificados en el rol.
5. Con la instancia en funcionamiento, el siguiente paso es clonar el repositorio de Git junto con el modelo necesario. Esto nos permite tener todo el código de nuestra aplicación disponibles a la instancia para poder empaquetar y crear el MAP.
6. Finalmente, subimos la imagen Docker a la Amazon Elastic Container Registry (ECR). Amazon ECR es un servicio de registro de contenedores completamente gestionado que facilita el almacenamiento, la gestión y el despliegue de imágenes Docker.

Una vez acabada el flujo de trabajo, podemos parar o eliminar la instancia, puesto que no la volveremos a utilizar. Cuando se para una instancia, esta deja de funcionar y no consume recursos de cálculo, pero los recursos de almacenamiento, como los volúmenes Elastic Block Store (EBS) asociados, se mantienen y se continúan aplicando cargos; la instancia se puede reiniciar en cualquier momento manteniendo su estado anterior, a pesar

de que la IP pública puede cambiar. En cambio, cuando se elimina una instancia EC2, esta se destruye completamente y deja de generar costes asociados.

Como que la clave RSA se ha guardado a la máquina virtual de GitHub, una vez finalizada el proceso, ya no tendremos acceso a esta clave y únicamente nos podremos conectar accediendo a la consola principal de AWS.

### 3.3.2 Configuración de la inferencia

El flujo de trabajo seguido para configurar la inferencia es el siguiente:

1. El primer paso es crear la instancia EC2, similar al proceso de creación del MAP. En este caso, en lugar de subir la imagen, la instancia descarga la imagen desde el registro ECR.

Al crear la instancia, ya le copio un script preconfigurat ec2.sh que se responsabiliza de descargar los datos desde el contenedor S3, ejecutar el modelo para realizar la inferencia, y finalmente, subir los resultados generados de nuevo a S3 y a HealthImaging. Por lo tanto, la función Lambda tan solo tiene que ejecutar este script.

Los scripts de configuración que se encuentran al repositorio tienen las variables vacías. Esto es así porque la información que contienen es confidencial. Cuando se ejecuta el workflow, la máquina virtual de GitHub sustituye el valor de estas variables de manera local utilizando los secretos configurados al repositorio. De este modo, se mantiene la seguridad y la confidencialidad de la información sensible. El código siguiente muestra como se realiza esta sustitución:

```
- name: Update ec2.sh with secrets
  run: |
    cd setup
    sed -i
    's|ACCOUNT_ID=""|ACCOUNT_ID="{{secrets.AWS_ACCOUNT_ID}}"|' ec2.sh
    sed -i
    's|ROLE_ARN=""|ROLE_ARN="{{secrets.AWS_MEDICALIMAGING_ROLE}}"|'
    ec2.sh
    sed -i
    's|DATASTORE_ID=""|DATASTORE_ID="{{secrets.AWS_DATASTORE_ID}}"|'
    ec2.sh
    sed -i 's|REGION=""|REGION="{{secrets.AWS_REGION }}"|' ec2.sh
    sed -i
    's|DOCKER_IMAGE_TAG=""|DOCKER_IMAGE_TAG="{{env.REPOSITORY}}:{{
    env.TAG }}"|' ec2.sh

- name: Send ec2.sh to EC2 instance
  run: |
    scp -i $SSH_PRIVATE_KEY_PATH -o StrictHostKeyChecking=no
    setup/ec2.sh $VM_NAME:/home/ubuntu/ec2.sh
    ssh -i $SSH_PRIVATE_KEY_PATH $VM_NAME "chmod +x
    /home/ubuntu/ec2.sh"
```

**Código 1.** Sustitución de las variables y envío del script a la instancia EC2

2. Una vez creada la instancia, se crea una función Lambda que será la encargada de iniciar la inferencia automáticamente. Esta función Lambda se encarga de gestionar la ejecución del modelo de IA en la instancia EC2 cuando se detecta un nuevo fichero a S3. Esto lo hacemos utilizando SSM, puesto que este permite enviar pedidos a la instancia EC2 para ejecutar scripts de manera remota.

```
# Zip the Lambda function code
zip -r function.zip lambda_function.py

# Create or update the Lambda function
aws lambda create-function --function-name $FUNCTION_NAME \
--zip-file fileb://function.zip --handler
lambda_function.lambda_handler \
--runtime python3.8 --role $ROLE_ARN --region $REGION \
|| \
aws lambda update-function-code --function-name $FUNCTION_NAME \
--zip-file fileb://function.zip --region $REGION

# Add S3 bucket permissions to invoke the Lambda function
aws lambda add-permission --function-name $FUNCTION_NAME --
statement-id S3Invoke --action lambda:InvokeFunction --principal
s3.amazonaws.com --source-arn arn:aws:s3:::$BUCKET_NAME
```

**Código 2.** creación de la función Lambda

La función en Python está en el repositorio de GitHub, pero se puede subir la función comprimida en un fichero ZIP, tal como se muestra en el código anterior. Esta práctica facilita la gestión y actualización de las funciones Lambda, permitiendo desplegar versiones nuevas simplemente actualizando el fichero ZIP y haciendo un llamamiento al API de AWS para crear o actualizar la función Lambda.

3. Se configura el servicio S3 para que envíe notificaciones a la función Lambda cada vez que se sube una nueva imagen. El fragmento de código de la siguiente página configura una notificación de acontecimientos en un bucket de S3 para invocar una función Lambda cuando se crea un nuevo objeto. Primero, se otorgan permisos a la función Lambda porque pueda ser invocada por S3. A continuación, se crea una configuración de acontecimientos en formato JSON, especificando que la función Lambda será invocada cuando se produzcan acontecimientos de creación de objetos al bucket, con un filtro opcional por solo considerar objetos con el prefijo "input/". Finalmente, se aplica esta configuración al bucket de S3 intermediando el API de S3, asegurando que cualquier nuevo objeto cargado que cumpla los criterios definidos desencadenará la ejecución de la función Lambda.

```

# Add S3 bucket permissions to invoke the Lambda function
aws lambda add-permission --function-name $FUNCTION_NAME --statement-
id S3Invoke --action lambda:InvokeFunction --principal
s3.amazonaws.com --source-arn arn:aws:s3:::$BUCKET_NAME

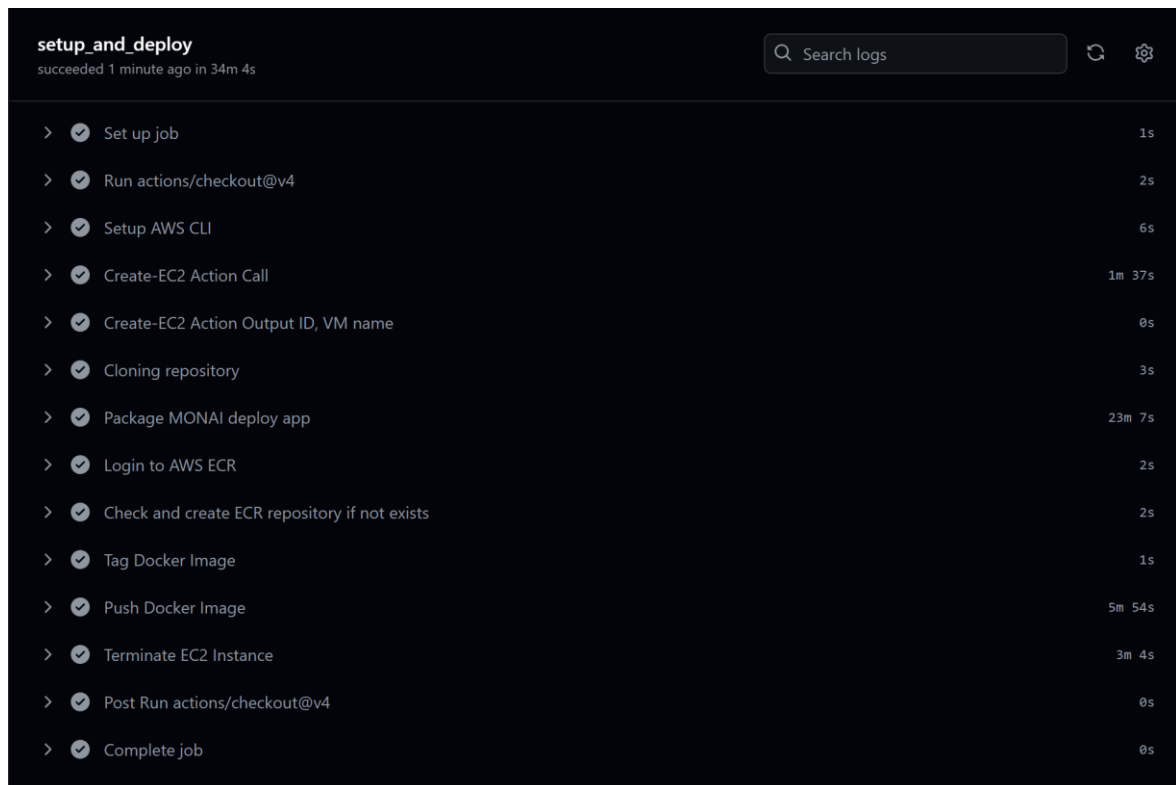
# Create S3 event configuration JSON
cat <<EOT > s3_event_configuration.json
{
  "LambdaFunctionConfigurations": [
    {
      "LambdaFunctionArn":
"arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME",
      "Events": ["s3:ObjectCreated:*"],
      "Filter": {
        "Key": {
          "FilterRules": [
            {
              "Name": "prefix",
              "Value": "input/"
            }
          ]
        }
      }
    }
  ]
}
EOT

# Apply the S3 event configuration
aws s3api put-bucket-notification-configuration --bucket $BUCKET_NAME
--notification-configuration file:///s3_event_configuration.json

```

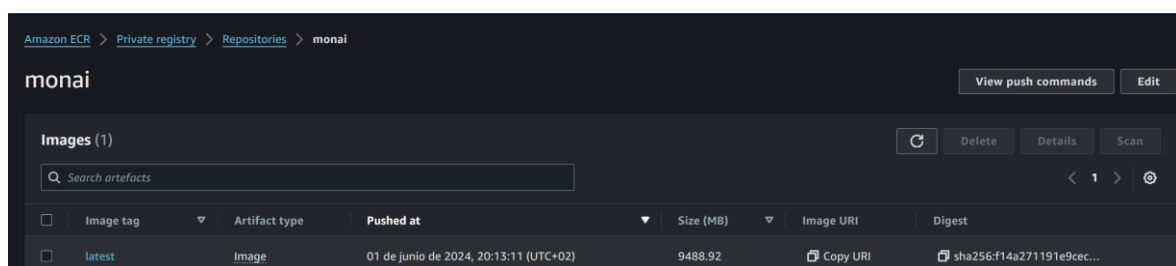
**Código 3.** Configuración de la notificación

## 4 Resultados



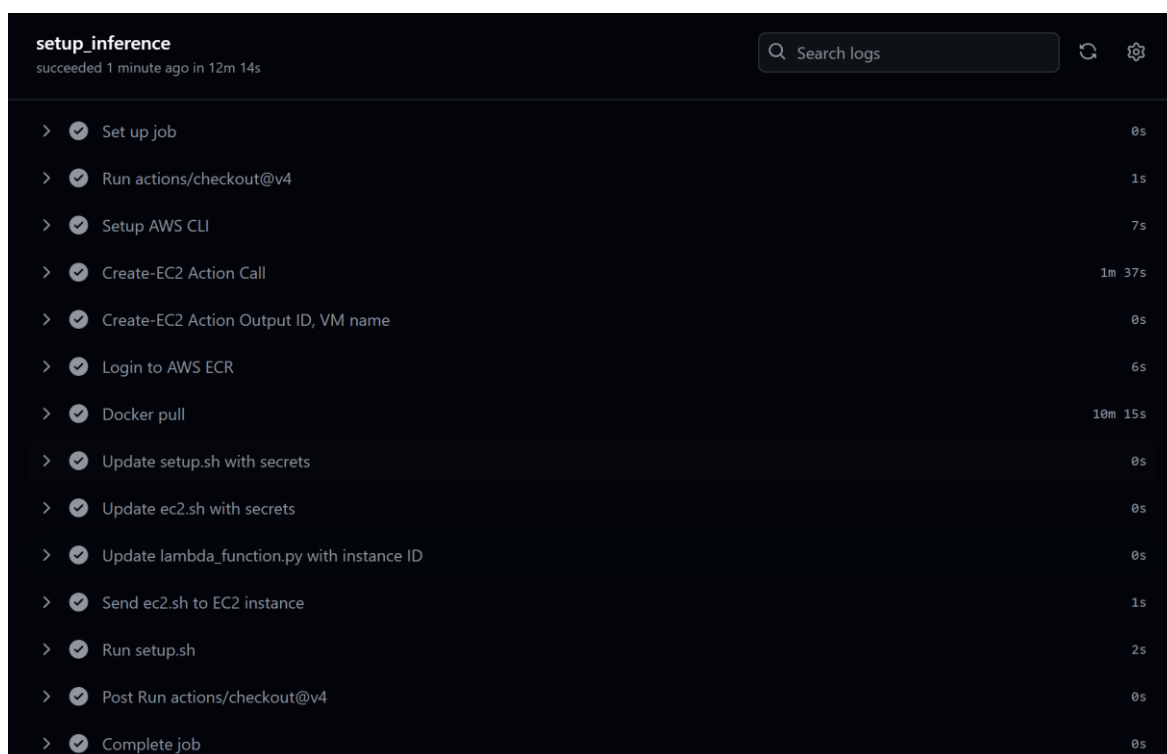
**Figura 11** Resultados de ejecuta create\_map.yml

Con estos resultados, podemos concluir que es correcto eliminar las instancias una vez finalizada el proceso, puesto que el tiempo que tarda a crear e iniciar la máquina es negligible en comparación con el tiempo que tarde a construir y subir el MAP. Como que solo tengo cuota de CPU para poner en marcha una única instancia, antes de crear una de nueva, tengo que eliminar o parar cualquier otra instancia que esté activa.



**Figura 12** Imagen subida a Amazon ECR

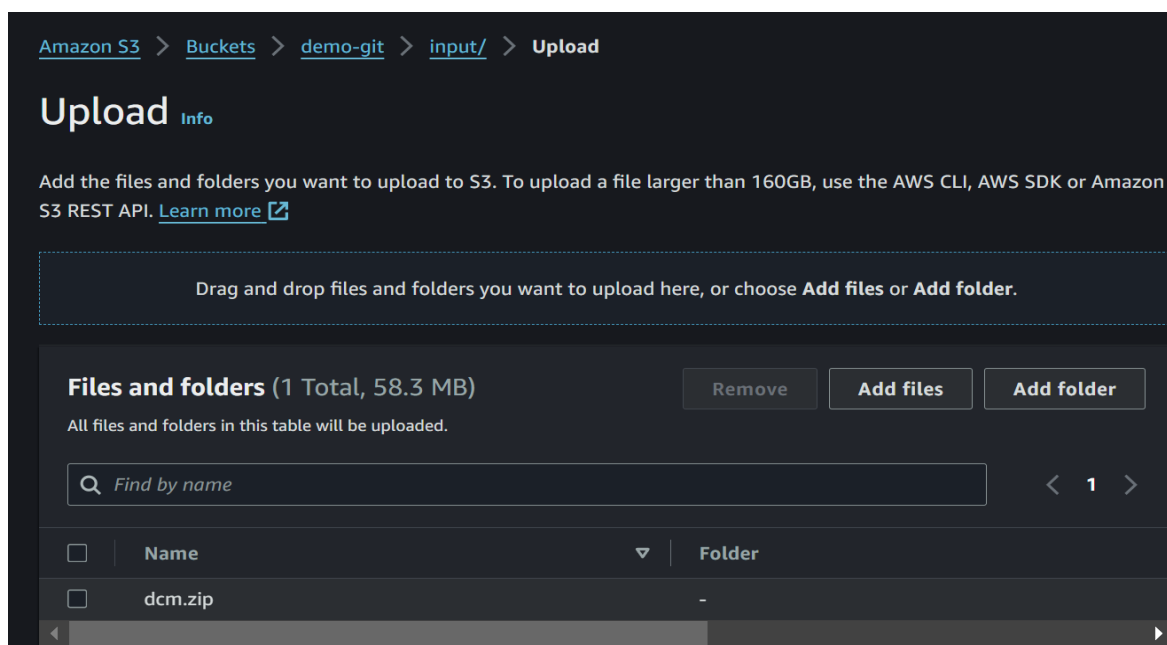
Todo este flujo de trabajo ha sido para crear la imagen y subirla al repositorio de Amazon ECR. Se puede observar que la imagen comprimida con el modelo y todos lo necesario para funcionar ocupa unas 9 GB.



**Figura 13** Resultados de ejecutar `configure_inference.yml`

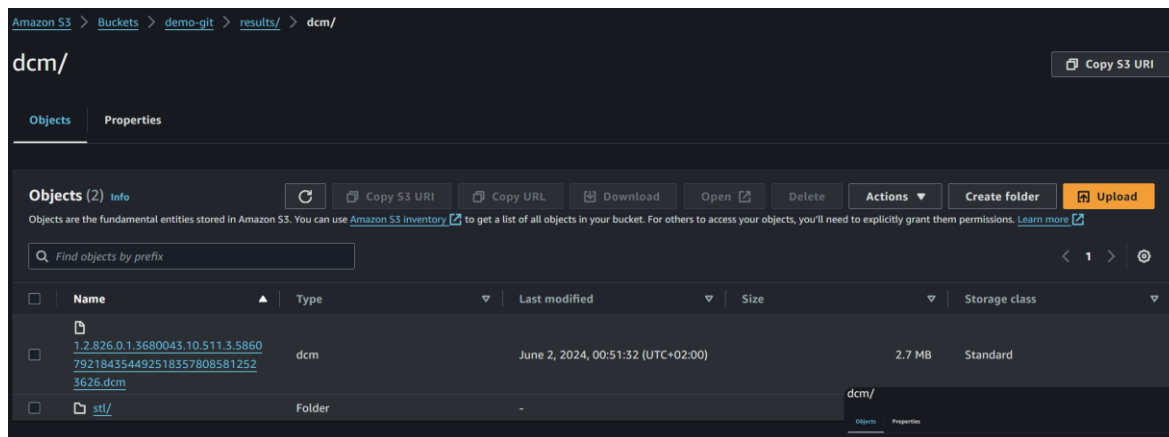
Configurar la inferencia es un proceso bastante rápido. Una vez puesta en marcha la máquina, se actualiza los ficheros (localmente dentro de la máquina virtual de github) con las variables secretas y se ejecuta `setup.sh` para crear la función lambda y conectarla con S3.

El paso que consume más tiempo es lo *docker pull*, que descarga la imagen Docker para tenerla preparada. Aunque esta configuración inicial descargue la imagen actual, si más adelante se actualiza el MAP, la máquina EC2 volverá a hacer un *docker pull* para obtener la versión más reciente, asegurando que siempre se utilice la última versión del modelo.



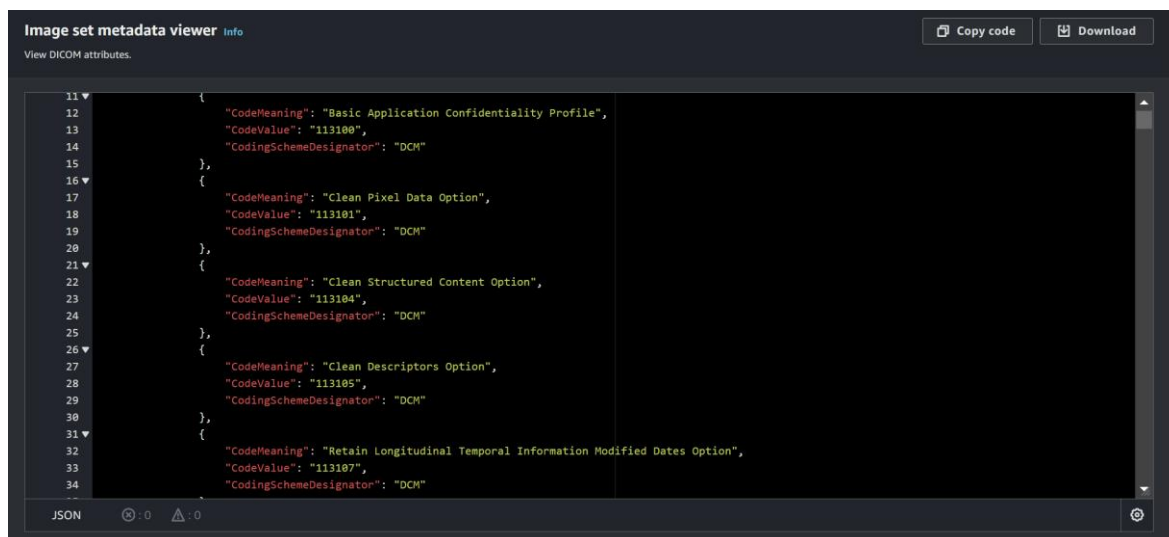
**Figura 14** Subir ficheros de prueba

Se han subido los ficheros de prueba directamente a S3, tal como se muestra a la imagen. En un escenario real, sería necesario desarrollar un frontend que permitiera a los usuarios interactuar de manera más fácil, intuitiva y segura, sin que los usuarios tuvieran control directo sobre S3.



**Figura 15** Resultados guardados en S3

Una vez colgamos un fichero de prueba (dcm.zip) en la carpeta input, vemos que nos ha guardado los resultados en una carpeta con el mismo nombre. En este modelo que hemos ejecutado el que nos genera es una imagen DICOM junto con un fichero .stl con la segmentación 3D.

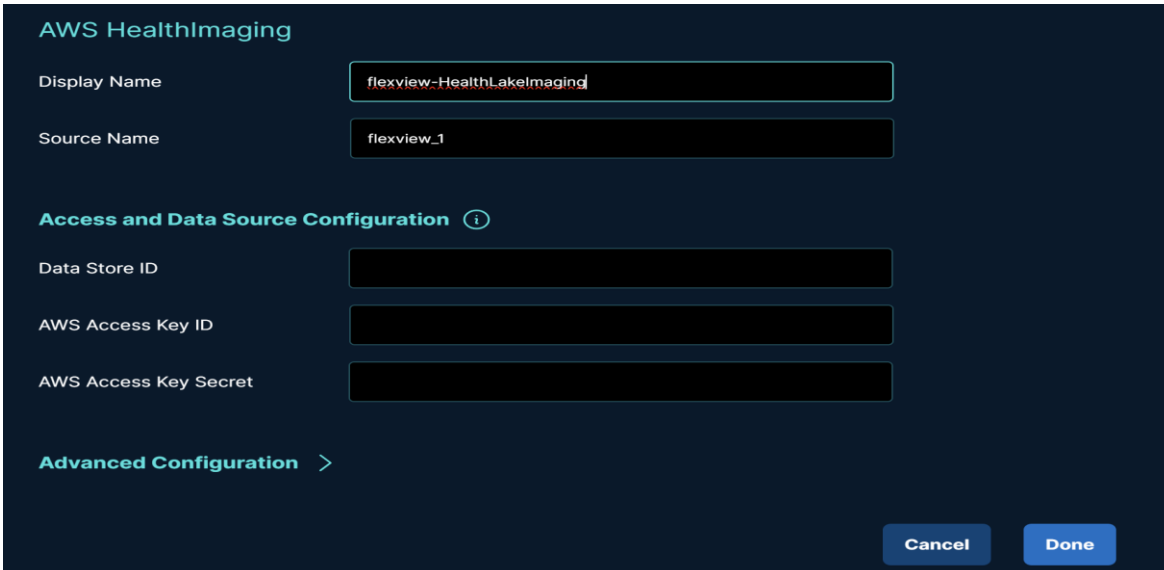


**Figura 16** Resultados en HealthImaging

Las imágenes también están subidas al HealthLake, pero solo podemos leer los metadatos de los DICOM. HealthImaging no permite visualizar las imágenes directamente, sin embargo, se podría crear una interfaz gráfica usando el código que podemos encontrar a: <https://github.com/aws-samples/aws-healthimaging-samples/tree/main/imaging-viewer-ui>

Esta interfaz permitiría a los usuarios interactuar de manera más efectiva con las imágenes médicas, mejorando su experiencia y facilitando el análisis y la interpretación de los datos visuales.

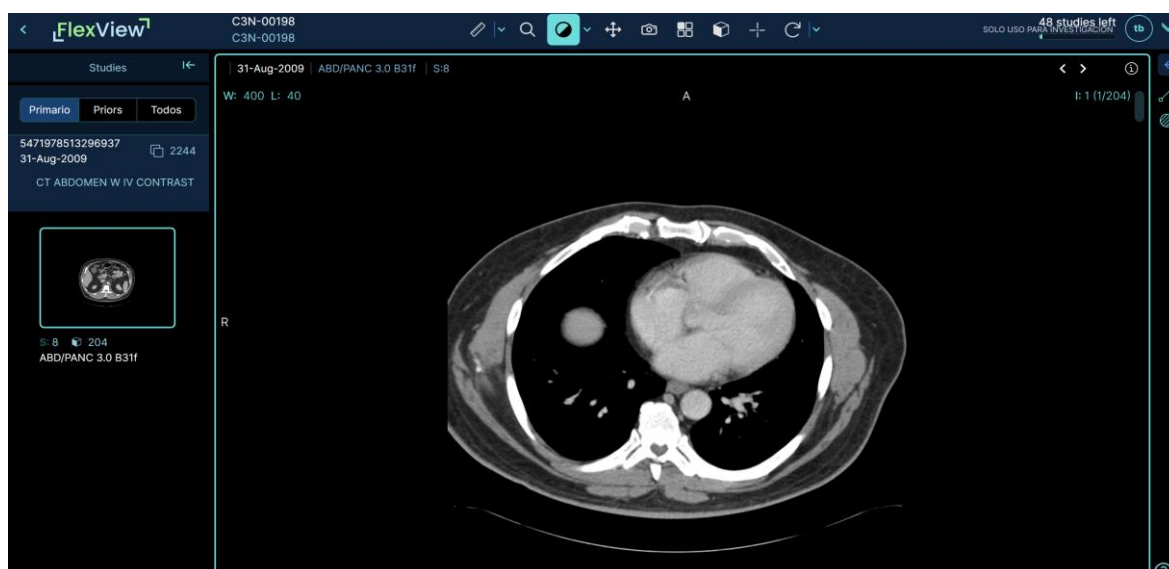
para hacerlo más fácil, podemos conectar el HealthImaging con FlexView, una herramienta que permite visualizar y gestionar imágenes médicas. FlexView ofrece la posibilidad de ver hasta 50 estudios únicos de manera gratuita. Una vez abiertos, estos estudios se pueden visualizar tantas veces como se quiera por cualquier usuario de la cuenta. Esto permite visualizar los DICOM sin necesidad de descargarlos ni tener ningún programa específico instalado al PC.



The screenshot shows the 'AWS HealthImaging' configuration window. It has a dark blue background with white text. At the top, it says 'AWS HealthImaging'. Below that, there are two input fields: 'Display Name' with the value 'flexview-HealthLakeImaging' and 'Source Name' with the value 'flexview\_1'. Underneath these is a section titled 'Access and Data Source Configuration' with a help icon. It contains three more input fields: 'Data Store ID', 'AWS Access Key ID', and 'AWS Access Key Secret'. At the bottom left, there is a link for 'Advanced Configuration' with a right-pointing arrow. At the bottom right, there are two buttons: 'Cancel' and 'Done'.

**Figura 17** Configuración FlexView

Para conectar FlexView con HealthImaging, solo hay que proporcionar el ID del Fecha Store y las credenciales de un usuario IAM con los permisos de lectura de HealthImaging para permitir la visualización de las imágenes DICOM.



**Figura 18** Visualización de un DICOM en FlexView



## 5 Evaluación de costos

El coste de realización del proyecto se ha basado principalmente en las horas que he dedicado y el coste de los servicios de AWS, puesto que GitHub y otras herramientas que he utilizado son gratuitas.

El tiempo total dedicado a la realización del proyecto es de aproximadamente 300 horas. Gran parte de este tiempo ha sido invertido a aprender a utilizar tres tecnologías con las cuales no había trabajado antes: AWS, GitHub Actions y MONAI. Inicialmente, dediqué varias horas a explorar la documentación y los tutoriales de cada una de estas herramientas para comprender como funcionan y qué eran las mejores prácticas para su implementación. Además, se hicieron varias pruebas para comprender el funcionamiento de los servicios y su integración. Por ejemplo, se probó la conexión de GitHub con S3 para automatizar la subida y gestión de archivos. A pesar de que esta conexión no se utilizó en el código final del proyecto.

Billing period <a href="#">Info</a>		Bill status <a href="#">Info</a>	
March 1 - March 31, 2024		🟢 Issued 04/02/2024	
Service provider		Total in USD	
Amazon Web Services EMEA SARL		USD 33.55	
Grand total:			USD 33.55

**Figura 19** Factura AWS febrero

La factura del mes de febrero fue debida a un descuido. No sabía muy bien cómo funcionaba el sistema y me dejé un notebook de SageMaker activo en la región de Londres (eu-west-2). Cuando entraba a SageMaker en la región Virginia (us-east-1) no veía ninguna instancia activa y, por lo tanto, no desactivé la instancia de Londres hasta después de unos cuantos días. A partir de esta lección, configuré una alarma que cada día me envía un correo electrónico si el día anterior gasté más de 0,01 dólares, esto me ha permitido evitar otras sorpresas y tener un mejor control sobre los gastos.

Dear AWS Customer,

You requested that we alert you when the **actual cost** associated with your *My Zero-Spend Budget* budget **exceeds \$0.01** per day. Yesterday, the **actual cost** associated with this budget is **\$1.10**. You can find additional details below and by accessing the AWS Budgets dashboard.

Budget Name	Budget Type	Budgeted Amount	Alert Type	Alert Threshold	ACTUAL Amount
My Zero-Spend Budget	Cost	\$1.00	ACTUAL	> \$0.01	\$1.10

**Figura 20** Ejemplo de notificación cuando se activa la alarma

Billing period <a href="#">Info</a>		Bill status <a href="#">Info</a>	
April 1 - April 30, 2024		🟢 Issued 05/02/2024	
Service provider		Total in USD	
Amazon Web Services EMEA SARL		USD 0.00	
Grand total:			USD 0.00

Figura 21 Factura AWS abril

En abril, a pesar de haber consumido recursos fuera de la capa gratuita de AWS, la factura total resultó ser de 0 dólares gracias a mí tutor, que me ayudó a conseguir un cupón de 100 euros para realizar el proyecto. Este apoyo fue esencial para poder utilizar los servicios necesarios sin generar costas adicionales, permitiéndome centrar en el desarrollo y ejecución del proyecto sin preocupaciones. El cupón me fue otorgado para participar en el programa Skills2Jobs, que premia los mejores trabajos finales de grado y másteres entre otros.

Billing period <a href="#">Info</a>		Bill status <a href="#">Info</a>	
May 1 - May 31, 2024		🟢 Issued 06/02/2024	
Service provider		Total in USD	
Amazon Web Services EMEA SARL		USD 8.31	
Grand total:			USD 8.31

Figura 22 Factura AWS mayo

En mayo, agoté el cupón de 100 euros que me ayudó a cubrir los gastos del proyecto. Como consecuencia, la factura del mes ascendió a 7,19 dólares. Este importe refleja el coste del uso de los servicios de AWS una vez agotada el crédito disponible.

En cuanto al coste de ejecutar la inferencia utilizando los servicios de AWS, a continuación se detallan los precios asociados a los diferentes servicios utilizados:

**Amazon S3:** tiene un coste de almacenamiento de 0,023 dólares de los Estados Unidos de América (USD) por GB. Las peticiones HEDE, COPY, POST y LIST tienen un coste de 0,005 USD por cada 1.000 peticiones, mientras que las peticiones GET, SELECT y otras peticiones cuestan 0,0004 USD por cada 1.000 peticiones.

**Amazon HealthImaging:** como parte del nivel gratuito de AWS, los nuevos clientes de AWS reciben 20 GB de almacenamiento en el mes, y 20.000 peticiones de APIO cada mes. Después de agotar el nivel gratuito, los costes son los siguientes:

- 0,105 USD per GB / mes para el nivel de acceso frecuente.
- 0,005 USD per cada 1.000 peticiones de API.

**ECR:** en cuanto al almacenamiento a Amazon Elastic Container Registry (ECR), el coste es de 0,10 USD por GB en el mes tanto para datos almacenados en repositorios públicos como privados.

**Lambda:** el nivel gratuito de AWS Lambda incluye un millón de solicitudes gratuitas en el mes y 400 000 GB/según de tiempos de computación en el mes. Por encima de este nivel, los costes son de 0,0000166667 USD por cada GB/segundo de tiempo de computación y 0,20 USD por un millón de solicitudes.

**EC2:** el coste de contratar una instancia g4dn.xlarge a AWS, que incluye una GPU NVIDIA T4, 4 vCPU, 16 GiB de memoria y 1 x 125 GB de almacenamiento SSD NVMe, es de 0,526 USD por hora en modo de pago bajo demanda. Si se reserva la instancia por un año, el coste efectivo se reduce a 0,316 USD por hora. En caso de reservarla por tres años, el coste efectivo es de 0,210 USD por hora.

El coste de ejecutar la inferencia es principalmente el tiempo que la instancia EC2 está activa. A pesar de que el almacenamiento de las imágenes en S3 y HealthImaging tiene un coste, este es muy inferior al de la instancia, puesto que las imágenes ocupan solo unos cuantos megabytes. El coste de almacenamiento a ECR no se considera aquí porque es un coste fijo independientemente de cuántas inferencias ejecutamos. Como que la imagen ocupa unas 10 GB, el coste mensual es de aproximadamente 1 USD.

En concreto, la inferencia tarda unos 90 según a completarse, y teniendo en cuenta que la instancia g4dn.xlarge tiene un coste de 0,526 USD por hora, el coste de ejecutar una inferencia es de 0,01315 USD, que redondeamos a 0,015 USD.

## 6 Legislación y protección de datos

El código fuente y el modelo que hemos utilizado como ejemplo son los códigos oficiales que se pueden encontrar en el GitHub de MONAI. Este código está disponible bajo la licencia Apache 2.0, lo cual nos permite utilizar, modificar y distribuir el código de acuerdo con los términos de la licencia. Esta licencia permite a los usuarios:

- **Uso:** Utilizar el código fuente en cualquier proyecto, incluyendo aplicaciones comerciales.
- **Reproducción:** Hacer copias del código para distribuirlas.
- **Modificación:** Hacer cambios en el código fuente para adaptarlo a necesidades específicas.
- **Distribución:** Redistribuir el código original o modificado a otros usuarios.
- **Sub-licencia:** Incluir el código en proyectos que se distribuyen bajo otras licencias.

Cuando utilizamos código fuente bajo la licencia Apache 2.0, tenemos que cumplir los siguientes requisitos para asegurar que se respetan los términos de la licencia:

- **Proporcionar una Copia de la Licencia:** Cada vez que redistribuimos el código, tenemos que incluir una copia de la licencia Apache 2.0.
- **Notificar las Modificaciones:** Si hacemos modificaciones al código original, tenemos que indicar claramente estos cambios, ya sea a través de comentarios en el código o mediante un registro de cambios.
- **Conservar Avisos de Derechos de Autor:** Tenemos que conservar todos los avisos de derechos de autor, patentes, marcas registradas y de atribución que se encuentran en el código original.
- **Incluir el Fichero NOTICE:** Si el código original incluye un fichero "NOTICE", tenemos que incluir una copia legible de los avisos de atribución contenidos en este fichero en nuestras obras derivadas. Esto se puede hacer de las siguientes maneras:
- **No Alterar la Licencia:** No podemos modificar la licencia original del código, pero podemos añadir nuestros propios términos de licencia para nuestras modificaciones u obras derivadas, siempre que estos términos no contradigan los términos de la licencia Apache 2.0.

En mi caso, he añadido el código fuente en mi repositorio de GitHub, pero no lo he modificado. Esto significa que estoy en las categorías de uso, reproducción y distribución del código. He incluido una copia original de la licencia Apache 2.0 en mi repositorio para cumplir con los requisitos de la licencia. Estos pasos aseguran que se respetan los términos de la licencia y que se proporciona la información necesaria sobre los derechos de autor del código original.

En cuanto a la legislación española, es necesario cumplir con la Ley Orgánica de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPD-GDD), el Reglamento General de Protección de Datos (RGPD) y la Ley de Servicios de la Sociedad de la Información y E-commerce (LSSI).

En este contexto, la LOPD-GDD y el RGPD son especialmente importantes, puesto que regulan la protección de los datos personales y establecen las obligaciones de los responsables del tratamiento de datos para garantizar los derechos de los usuarios. Esto incluye aspectos como la seguridad de los datos, el consentimiento de los usuarios, la transparencia en el tratamiento de los datos y la notificación de violaciones de seguridad.

Por otro lado, la LSSI no aplica en nuestro caso porque únicamente estamos ejecutando el modelo a la nube y no estamos haciendo operaciones económicas por internet. La LSSI obliga a cumplir todas aquellas personas físicas o jurídicas que efectúen operaciones económicas por internet, como por ejemplo páginas web que obtengan ingresos de forma directa (venta de productos o servicios) o indirecta (publicidad en línea, banners, patrocinios, etc.). Por lo tanto, como que no encajamos en estas categorías, no nos es aplicable esta legislación.

La protección de datos médicos es un aspecto crítico en el desarrollo y despliegue de aplicaciones que gestionan información de salud. Los datos médicos son especialmente sensibles porque contienen información personal y confidencial sobre la salud y el bienestar de los individuos. Esto hace que la protección de estos datos sea fundamental para garantizar la privacidad y seguridad de los pacientes.

En nuestro caso, los datos médicos utilizados en el proyecto son anonimizados y públicas. Esto significa que los datos han sido transformados de forma que no se puede identificar directamente a los pacientes, pero permitiendo el tratamiento para finalidades legítimas como la investigación y el desarrollo de modelos de inteligencia artificial.

Anonimizar los datos implica eliminar o modificar los datos personales que podrían identificar directamente a una persona, como por ejemplo nombres, direcciones, números de identificación o cualquier otra información que pueda ser utilizada para identificar al individuo. Pseudonimitzar los datos, por otro lado, implica reemplazar información identificativa con pseudónimos o códigos, que pueden ser revertidos bajo ciertas condiciones estrictas, pero no son directamente identificables.

En cuanto a los ficheros DICOM (Digital Imaging and Communications in Medicine), que son utilizados ampliamente en imágenes médicas, estos pueden contener datos personales en sus metadatos. Sin embargo, en este proyecto, los ficheros DICOM utilizados han sido anonimizados, eliminando cualquier información que pueda ser utilizada para identificar a los pacientes. Esto asegura que los datos puedan ser compartidos y procesados de manera segura, cumpliendo con las normativas de protección de datos y garantizando la privacidad de los individuos.

A continuación, se muestra un ejemplo de cómo se presentan los datos en las imágenes que hemos usado:

"EthnicGroup": "8"

"PatientBirthDate": null

"PatientID": "C3N-00198"

"PatientIdentityRemoved": "YES"

"PatientName": "C3N-00198"

"PatientSex": "M"

En este ejemplo, el nombre del paciente y el identificador del paciente han sido reemplazados por un código ("C3N-00198"), y se ha indicado claramente que la identidad del paciente ha sido eliminada ("PatientIdentityRemoved": "YES"). La fecha de nacimiento del paciente ha estado nula ("PatientBirthDate": null) y otros datos como el grupo étnico y el sexo del paciente se mantienen, pero sin la posibilidad de relacionarlos con la identidad real del paciente.

En resumen, a pesar de que trabajamos con datos médicos, la información es totalmente anonimizada, y por tanto no se puede identificar a ninguna persona a partir de los ficheros DICOM utilizados en nuestro proyecto. Esta práctica es fundamental para respetar la privacidad de los pacientes y cumplir con las regulaciones de protección de datos.

## 7 Conclusiones

En conclusión, este proyecto ha demostrado que la prueba de concepto es viable y que se puede automatizar el proceso de despliegue de modelos de inteligencia artificial para la inferencia en imágenes médicas utilizando servicios a la nube de AWS. Mediante el uso de GitHub Actions para gestionar y automatizar los procesos de construcción, prueba y despliegue, se han obtenido buenos resultados, reduciendo significativamente el tiempo y los esfuerzos necesarios para llevar los modelos desde la fase de desarrollo hasta la producción. Hay que destacar que este proyecto se ha centrado principalmente en el backend, y se ha utilizado Amazon S3 para la gestión de los ficheros. En uno en torno a producción real, sería necesario desarrollar una interfaz que permita a los usuarios subir imágenes y acceder a los resultados de una manera más intuitiva y segura.

La parte más complicada del proyecto ha estado entender como ligar todos los servicios que ofrece Amazon para conseguir una integración eficiente y funcional. La configuración de los permisos de los roles y usuarios ha estado particularmente problemática. Me he encontrado a menudo con errores indicando que no tenía acceso, como por ejemplo: "User: arn:aws:iam::\*/ is not authorized to perform: ...". Estos errores se han presentado repetidamente durante el desarrollo, requiriendo un tiempo considerable para ajustar y verificar los permisos correctos.

Esta dificultad se deriva de la necesidad de garantizar la seguridad ofreciendo los mínimos privilegios necesarios para cada rol/usuario, pero al mismo tiempo asegurando que los servicios puedan interactuar correctamente. Esta gestión de permisos es crucial para prevenir accesos no autorizados y mantener la integridad del sistema, pero su configuración precisa puede ser un reto significando. Esto ha puesto de manifiesto la complejidad de gestionar de manera segura los servicios a la nube y la importancia de comprender en detalle la configuración de permisos de AWS para evitar problemas de autenticación y autorización.

A pesar de que quería hacer el proceso el más autónomo posible, hay algunos aspectos que todavía requieren intervención manual por parte del usuario. En concreto, el usuario tiene que crear manualmente los roles IAM necesarios para el funcionamiento de la aplicación (solo la primera vez para configurar los secretos de GitHub). Permitir que un usuario IAM (GitHub) tenga el poder de crear y gestionar roles puede llevar a una mala configuración de permisos, lo cual podría comprometer la seguridad de todo el sistema. Además, este nivel de acceso podría ser explotado para ejecutar acciones no autorizadas si las credenciales del usuario se ven comprometidas. Por lo tanto, he optado para mantener esta tarea como una responsabilidad manual, asegurando así que el control de los permisos y roles se mantenga bajo supervisión directa y consciente del administrador del sistema.

Realizando este proyecto, he aprendido muchas cosas nuevas, incluyendo el uso de GitHub Actions y la gestión de servicios en la nube con AWS. Este proyecto me ha ayudado a comprender mejor como funciona el cloud computing, sus capacidades y ventajas, así como la importancia de tener cura por no descuidar-se y evitar sorpresas en la factura. Esta experiencia ha estado muy valiosa y enriquecedora, proporcionándome una base sólida para futuros proyectos en el campo de la inteligencia artificial y la nube.

## 8 Referencias

- [1] Los modelos de IA raramente llegan a producción: <https://www.kdnuggets.com/2022/01/models-rarely-deployed-industrywide-failure-machine-learning-leadership.html> (visitado 25-05-2024)
- [2] Usar MONAI en AWS (Vídeo): <https://www.nvidia.com/en-us/on-demand/session/gtcspring23-se52191/> (visitat 23-03-2024)
- [3] Por qué los proyectos de IA no llegan a producción: <https://medium.com/@jvanlooy/7-reasons-why-80-of-ai-projects-never-make-it-to-production-5e25c2ad4d46> (visitado 25-05-2024)
- [4] Por qué los proyectos de IA no llegan a producción <https://d2iq.com/blog/why-87-of-ai-ml-projects-never-make-it-into-production-and-how-to-fix-it> (visitado 25-05-2024)
- [5] Copiar ficheros de S3 a EC2: <https://stackoverflow.com/questions/64159874/copying-s3-file-to-ec2-every-time-file-posted-to-bucket> (visitado 05-05-2024)
- [6] Desplegar MONAI en AWS (vídeo): <https://www.youtube.com/watch?v=ALuHhMYuWFk&t=1527s> (visitado 15-03-2024)
- [7] Repositorio Github MONAI en AWS: <https://github.com/aws-samples/monai-on-aws-workshop>
- [8] Repositorio Github MONAI deploy: <https://github.com/Project-MONAI/monai-deploy-app-sdk>
- [9] Documentación Terraform: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs>
- [10] Documentación Github Actions: <https://docs.github.com/en/actions>
- [11] Documentación AWS: [https://docs.aws.amazon.com/es\\_es/](https://docs.aws.amazon.com/es_es/)
- [12] Documentación MONAI: <https://docs.monai.io/projects/monai-deploy-app-sdk/en/latest/>



## Apéndice

### A Configuración AWS

#### Rol EC2

Accede a la consola de IAM y selecciona "Roles" al menú lateral e hiciera clic a "Create role". Selecciona "AWS service" como tipo de entidad de confianza y elige "EC2" como caso de uso específico, puesto que quieres permitir que las instancias EC2 asuman este rol.

The screenshot shows the 'Create role' wizard in the AWS IAM console. On the left, there are three steps: Step 1 (Select trusted entity), Step 2 (Add permissions), and Step 3 (Name, review, and create). The main area is titled 'Select trusted entity' and contains two sections. The 'Trusted entity type' section has four options: 'AWS service' (selected), 'AWS account', 'Web identity', and 'SAML 2.0 federation'. The 'Use case' section has a dropdown menu with 'EC2' selected. Below the dropdown, it says 'Choose a use case for the specified service. Use case: EC2'. At the bottom, it says 'Allows EC2 instances to call AWS services on your behalf'.

En este paso, añade las políticas de permisos necesarias al rol para garantizar que el rol tenga las autorizaciones adecuadas para interactuar con otros servicios de AWS. Búsqueda y selecciona las siguientes políticas gestionadas por AWS:

- *AmazonEC2ContainerRegistryFullAccess*: Permite acceder completamente al registro de contenedores de Amazon.
- *AmazonS3FullAccess*: Permite acceder completamente a los recursos de Amazon S3.
- *AmazonSSMFullAccess*: Permite acceder completamente a Amazon SSM, necesario para enviar pedidos a la instancia EC2.
- *AWSHealthImagingFullAccess*: Permite acceder completamente a los recursos de Amazon HealthImaging.

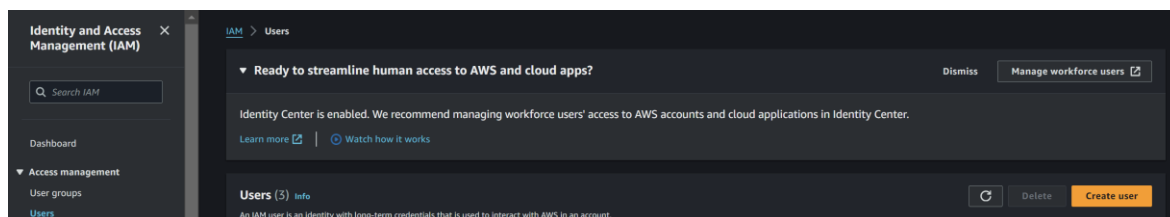
The screenshot shows the 'Permissions policies' page in the AWS IAM console. It has tabs for 'Permissions', 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions'. The 'Permissions' tab is active. The page title is 'Permissions policies (4)' and it says 'You can attach up to 10 managed policies.' There is a search bar and a 'Filter by Type' dropdown set to 'All types'. Below is a table with four rows, each representing a policy.

	Policy name	Type	Attached entities
<input type="checkbox"/>	AmazonEC2ContainerRegistryFullAccess	AWS managed	3
<input type="checkbox"/>	AmazonS3FullAccess	AWS managed	8
<input type="checkbox"/>	AmazonSSMFullAccess	AWS managed	3
<input type="checkbox"/>	AWSHealthImagingFullAccess	AWS managed	3

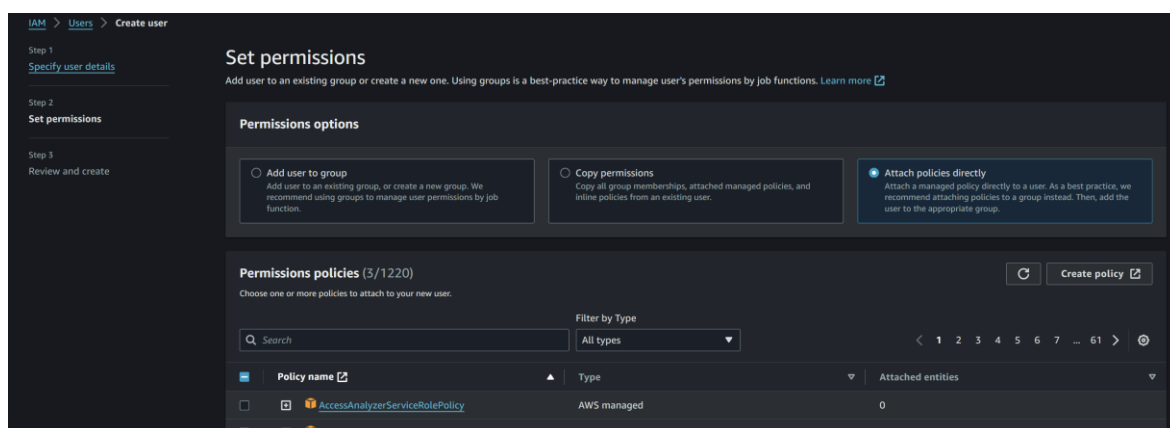
Revisa las configuraciones para asegurarte que todo esté configurado correctamente. Finalmente, crea el rol asignándole un nombre que refleje su uso y función dentro del proyecto.

## Usuario IAM

Ve a la consola de AWS y selecciona el servicio IAM (Identity and Access Management). Selecciona "Users" e hiciera clic a "Create user". Asigna un nombre al usuario, por ejemplo, github-action.



Selecciona "Attach policies directly" para añadir políticas de permisos. Añadís políticas gestionadas por AWS como AmazonEC2FullAccess, AmazonS3FullAccess, AWSLambda\_FullAccess.



Clica el botón de “Create policy” y modificáis el JSON que encontrarás por defecto por el siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:AssociateIamInstanceProfile"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::${ACCOUNT_ID}:role/${EC2_ROLE}"
    }
  ]
}
```

Esta política permite al usuario describir instancias de EC2 y asociar un perfil IAM a estas instancias. También permite pasar un rol específico a la instancia EC2. Esto es más seguro que dar acceso completo a IAM, puesto que solo permite las acciones necesarias.

Después de crear el usuario, selecciona el usuario reciente creado a la lista de usuarios. Ve a la pestaña "Security credentials" e hiciera clic a "Create access key". Guarda el "Access key ID" y el "Secreto access key" que se muestran en los secretos de github.

The screenshot shows the AWS IAM console. The top navigation bar includes tabs for Permissions, Groups, Tags, Security credentials, and Access Advisor. The main content area is titled "Permissions policies (4)" and shows a list of policies attached to the user. Below this, the "Retrieve access keys" section is visible, showing the "Access key" and "Secret access key" fields. The "Access key" field is redacted with a black box, and the "Secret access key" field is also redacted with a black box, with a "Show" link next to it.

Policy name	Type	Attached via
AmazonEC2FullAccess	AWS managed	Directly
AmazonS3FullAccess	AWS managed	Directly
Assign_EC2_Role	Customer managed	Directly
AWSLambda_FullAccess	AWS managed	Directly

**Retrieve access keys**

**Access key**  
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key: [Redacted]  
Secret access key: [Redacted] [Show](#)

## HealthImaging

A la consola de gestión de AWS, dirígete a la sección "AWS HealthImaging". Fez clic a "Create fecha store". Introduce un nombre para tu fecha store (por ejemplo, monai-HealthImaging). Configura las opciones necesarias e hiciera clic a "Create".

The screenshot shows the AWS HealthImaging console. The left sidebar has a "Data stores" section. The main content area is titled "AWS HealthImaging" and includes a "Get started" section with a "Create data store" button. Below this, the "Data stores (1)" section is visible, showing a table with one data store named "monai-HealthImaging".

**AWS HealthImaging**  
Store, analyze, and share medical images in the cloud at petabyte scale

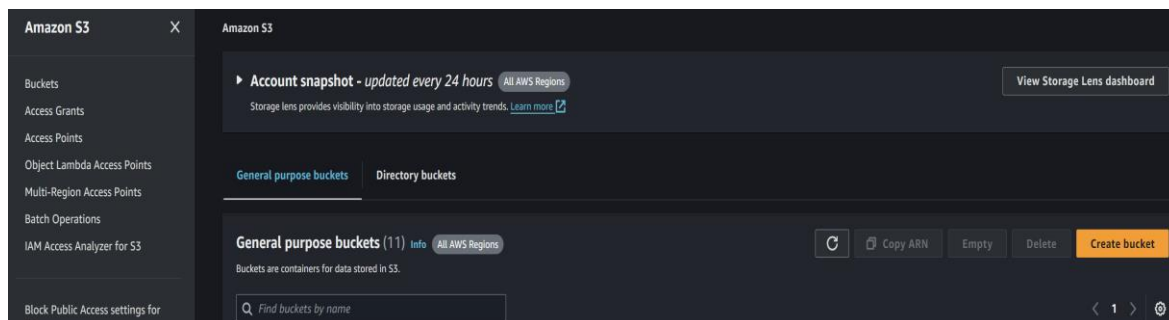
**Get started**  
Store, analyze, and share medical images in the AWS cloud at sub-second speed. Get started by creating a data store.  
[Create data store](#)

**Data stores (1)**

Name	Data store ID	Status	Created (UTC-5:00)
monai-HealthImaging	[Redacted]	Active	May 20, 2024 04:06:47

## Bucket S3

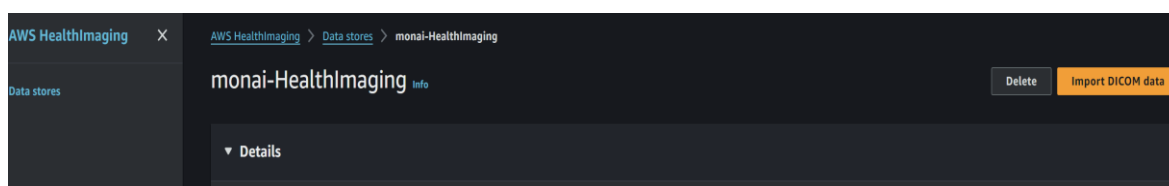
Inicia sesión a la consola de AWS y navega a la sección de S3. En la página principal de S3, hiciera clic a "Create bucket" para iniciar el proceso de creación de un nuevo bucket. Asigna un nombre único al bucket y selecciona la región donde quieres almacenar los datos. Configura otras opciones como la configuración de permisos, la configuración del versionado y las etiquetas si es necesario.



## Rol HealthImaging

Para crear el rol IAM necesario para el uso con AWS HealthImaging, puedes seguir la guía detallada proporcionada por AWS. La guía cubre los pasos necesarios para crear y configurar un rol IAM que permita el acceso seguro y adecuado a los servicios de HealthImaging. Puedes acceder a la guía [aquí](#).

Opcionalmente: Una vez creada el Fecha Store, puedes importar datos DICOM directamente a HealthImaging. Fez clic a "Importe DICOM fecha" para empezar. Sigue las instrucciones para seleccionar los datos DICOM que quieres importar. Este proceso creará automáticamente un nuevo rol que podrás utilizar.



***Rol lambda***

Accede a la consola de IAM y selecciona "Roles" al menú lateral e hiciera clic a "Create role". Selecciona "AWS service" como tipo de entidad de confianza y elige "Lambda".

Asigna permisos para S3 y SSM y clic el botón de "Create policy" y modificáis el JSON que encontrarás por defecto por el siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Start*",
        "ec2:Stop*"
      ],
      "Resource": "*"
    }
  ]
}
```

Esta política IAM permite al rol asociado crear grupos de logs, flujos de logs y enviar acontecimientos de logs a AWS CloudWatch Logs, así como iniciar y parar instancias EC2. Los permisos están diseñados para ser bastante amplios para permitir el uso de estos servicios en todos los recursos y regiones de AWS.

## B Configuración GitHub

Para configurar los secretos del repositorio a GitHub, primero tenéis que ir a vuestro repositorio y hacer clic a la pestaña Settings a la parte superior derecha. A continuación, a la barra lateral izquierda, seleccionáis Secretos and variables después hacéis clic a Actions. Después, hacéis clic al botón verde New repository secreto a la parte superior derecha de la página para añadir un nuevo secreto. Introducís el nombre y el valor del secreto necesario. Repetís este proceso para cada secreto necesario. Este proceso asegura que las credenciales y otras informaciones delicadas se gestionan de manera segura, siendo encriptadas antes de ser almacenadas y desencriptadas solo al momento del uso en los workflows de GitHub Actions.

